U.S. DEPARTMENT OF COMMERCE
National Technical Information Service

PB-281 248

EPA Executive Seminar on Database Management Systems Held on October 25-26, 1977

National Computer Center, Research Triangle Park, N C

1977

PB 281 248

NATIONAL COMPUTER CENTER

Research Triangle Park

NORTH CAROLINA

EPA

EPA EXECUTIVE SEMINAR ON DATABASE MANAGEMENT SYSTEMS

Edited by:
Dr. David F. McAllister
Computer Science Department
North Carolina State University
Raleigh, North Carolina 27650

REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U. S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

OCTOBER 25-26, 1977

98

FOREWORD

The U. S. Environmental Protection Agency was established December 2, 1970, as the result of a Presidential Order. The action reflected a public commitment to the control and abatement of pollution. For the first time, 15 environmental control programs, formerly scattered through the Federal Government, were unified in a single, independent agency.

EPA is charged with the basic mission of mounting an integrated, coordinated attack on the environmental problems of air and water pollution, solid waste management, pesticides, radiation, and noise.

Above all else, EPA is a regulatory agency with responsibilities for the establishment and enforcement of environmental standards as specified in statutes enacted by Congress.

To ensure that the Agency is responsive to environmental problems throughout the country, Regional Offices are located in ten major cities. Each Regional Office is staffed by pollution-control specialists and headed by a Regional Administrator with broad authority to act for EPA in matters within that Region's jurisdiction.

On New Year's Day, 1970, nearly a year before the creation of EPA, the will of the American people regarding the need for ecological controls was formally expressed when the National Environmental Policy Act was signed by the President, thereby establishing a national priority to "maintain conditions under which man and nature can exist in productive harmony."

Since then, new environmental laws have been passed at every level of American government, demonstrating a pervasive concern for protection of human health and life-sustaining ecosystems.

EPA is responsible for administering Federal laws on environmental control. It helps to protect the environment in a number of ways:

1. Standard Setting and Enforcement

The key to sensible environmental control lies in the determination of what changes in our environment are tolerable or useful and what changes must be limited because they are harmful. Such decisions must be based on sound technical information, as well as a keen regard for the interests of our people.

Recognizing this, Congress has passed several laws requiring EPA to define the levels which certain environmental pollutants must not exceed. EPA has developed three separate but interrelated processes in connection with this mandate. These include determining specific environmental levels for several pollutants, enforcement activities, and diversified monitoring programs maintained by State and local governments.

a. Standard Setting

EPA is charged with the responsibility of ensuring that those standards it does set will be sufficient to protect the public health and welfare, whether they involve restricting pesticide use or setting emission levels for automobiles. State and local governments may develop additional controls or programs for various reasons, but EPA's direct responsibilities are restricted to protecting health and welfare.

b. Enforcement Programs

EPA's philosophy has been to encourage voluntary compliance by private industry and communities or to encourage State and local governments to perform whatever enforcement activities are needed to meet EPA standards. If these agencies fail to produce effective plans for pollution abatement or if they don't enforce the programs they do develop, EPA must do so under enforcement provisions contained in most of the major environmental laws passed by Congress.

c. Monitoring Programs

Several kinds of monitoring processes and activities exist within EPA. Some are broadly based monitoring programs that determine whether pollution levels and emissions are increasing or declining. Others determine if the various abatement programs developed by EPA and State and local governments are as effective as they should be.

2. Research and Development

Effective environmental action as directed by Federal legislation requires precise technical data on possible threats to health and the environment posed by the various substances which are introduced into the biosphere. The research and development arm of EPA supports the Agency's primary functions of developing and enforcing appropriate regulations and standards by providing such data.

EPA's research program is authorized under various major Congressional acts. This legislation allots more than one-fifth of the Agency's operating budget for scientific study. These funds support the activities of a technical and support staff who work at EPA's Washington headquarters, in 15 major laboratories and in a number of smaller field stations, and those in research programs carried on outside of EPA through grants and contracts with academic, research, and industrial communities, and through cooperative agreements with other Federal, State, and local agencies.

The research and development program centers on four principal areas. The Office of Monitoring and Technical Support is responsible for: (a) developing reference or standard environmental measurement and monitoring equipment, techniques, and systems; (b) developing Agency-wide quality assurance programs; (c) disseminating scientific and technical knowledge; and (d) providing technical support, including monitoring and analytical support, to the Agency. The Office of Health and Ecological Effects studies the implications of varying types and levels of pollution on human health and the environment. The Office of Energy, Minerals, and Industry is responsible for assessing, developing, and demonstrating pollution control technology to mining and industrial activities. This Office also plans and administers a comprehensive Federal energy and environmental research, development, and demonstration program.

The Office of Air, Land, and Water Use is responsible for planning a comprehensive research program on water supply and on municipal wastes, including waste water, and solid and hazardous waste management. This office also is responsible for research on: the transportation and disposition of pollutants in the environment; areawide environmental management; and pollution from non-point sources such as urban run-off and agricultural and forestry activities.

Here are some examples of projects being conducted by EPA's Office of Research and Development:

- Investigation of the relationship between human health effects and exposure to combinations of pollutants reaching man primarily in air. Findings from such work will be directly related to evaluating existing standards and the need for new ones.

- An assessment of the incidence of illness in bathers at relatively clean and relatively polluted beaches is being conducted to determine how illness can be correlated to the water quality. The information gained from the research will be used to help develop health criteria for recreational water quality.

- EPA's Advanced Waste Treatment Program in Cincinnati is investigating new methods of purifying waste water. This research eventually may lead to the reuse of water, not only for industrial purposes but also for domestic use. Methods of sewage sludge disposal and use also are being investigated.

- The Regional Air Pollution Study (RAPS) in St. Louis is the largest, most complex air pollution study ever made. EPA laboratories are attempting to develop more precise predictive models of pollutant formation and transport so that air pollutant effects can be linked with sources and thus better pollution controls can be devised.

- EPA has been developing and demonstrating flue gas desulfurization technology, including the device commonly known as the stack gas "scrubber." Scrubbers can be used to control sulfur dioxide and particulate matter from coal-fired electric power plants.

- Technical and scientific findings are published and distributed to help advance the total body of scientific knowledge on pollution effects, pollution control technologies, and monitoring methods.

EPA's "Technology Transfer" program has the objective of helping to avoid a large national investment in obsolete pollution control technologies. The program is designed to bridge the gap between research and development and the general use of new or improved technologies.

3. Financial and Technical Assistance

By providing financial and technical assistance to State, regional, and local jurisdictions, EPA serves as a catalyst for environmental protection efforts at all levels of government. EPA grants Federal funds for the construction and operation of various types of facilities to reduce pollution. It also demonstrates new pollution control technology.

EPA's ten Regional Offices provide assistance to State and local authorities, industries, and citizens to help them solve environmental problems.

Automatic Data Processing within EPA

Consistent with the decentralized organization of EPA, the data processing functions are also decentralized. Application development is performed by the programs which use the resources provided by a central services group. The data processing services is centered around two major processing centers. The Washington Computer Center (IBM) and the National Computer Center (UNIVAC) provide most of the ADP resource for EPA.

Over the past four years these centers have provided a database management system to their users. System 2000 provided by the MRI Corporation in Austin, Texas was selected due to its ability to operate on both IBM and UNIVAC equipment. System 2000 was installed in both of these centers and made available to the user community as a utility. Typical System 2000 training as provided by MRI was offered to any user who desired to take it. Very little effort was made to emphasize or to deemphasize the utilization of System 2000.

During the four years that System 2000 has been available in EPA, it has been used for over 30 application developments, some of which have been successful and several which have failed. In most cases the failures resulted from a lack of understanding of database management principles. The successes were created by a more careful and selective approach to the use of System 2000.

In early 1977 the Director of the Management Information and Data Systems Division, Mr. Willis Greenstreet, directed that EPA should take a more closer look at its use of database management systems and determine if a new course should be plotted. In evaluating the status of DBMS in EPA it was determined that EPA middle and upper management had not been privileged to any training on Database Management Systems.

As part of an overall development plan for DBMS within EPA, several actions were taken:

1. A comprehensive design effort was started to develop a system design for an agency personnel system. The design team was instructed to develop the system design consistent with database management needs.

2. The National Computer Center installed a CODASYL structured database system (DMS 1100) provided by Sperry Rand.

3. A prototype personnel database system was developed to both demonstrate DMS 1100 and to demonstrate the advantages of DBMS to EPA management.

4. An executive seminar to acquaint EPA mid and upper management on database management systems was planned.

The purpose of the executive seminar was to both create interest in database management systems and to also warn about the dangers of improper use. Vendor presentations on the great advantages of a particular database management system would be inappropriate but a scholarly approach to database management systems including their uses and problems would be in order. The collection of papers included in this document were delivered at this seminar.

LC

of the **EPA EXECUTIVE SEMINAR**

begin

FOR: EPA Executives (AA's, DAA's and Division Directors)

The **OBJECTIVE:** LC To acquaint EPA management with database management systems (DBMS) and how they can help the Federal Manager,

SPONSOR: Management Information and Data Systems Division of EPA

WITH ASSISTANCE OF: Computer Science Department, N. C. State University and the National Computer Center, EPA

COORDINATION BY: The Division of Continuing Education, N. C. State University

PROGRAM

Topics included the following

Monday p.m.

7:00- 9:00 - Social for Early Arrivals

Tuesday a.m.

8:30- 9:00 - Coffee and Danish

9:00-10:15 - Speaker: M. H. Schwartz
Topic: "What can DBMS do for the Federal Manager",

10:15-10:30 - Break - Coffee and Soft Drinks

10:30-11:45 - Speaker: Leo J. Cohen. . . . . p. 1
Topic: "Organization Considerations for Successful DBMS"

Tuesday p.m.

Luncheon

12:00- 1:00 - Welcome by R. E. Johns, Director of the N. C. Office of State Management Systems

1:15- 2:30 - Speaker: Dr. James D. Powell . . . . p. 17
Topic: "Case Studies of DBMS",

2:30- 2:45 - Break - Coffee and Soft Drinks

2:45- 3:45 - Panel Discussion with All Speakers. . . . . p. 28

4:00- 5:00 - NCC Demonstrations in the Council Room

6:30- 7:30 - Social

7:30- 9:00 - Banquet
Speaker - Sam D. Bundy, Member of the North Carolina State Legislature

Wednesday a.m.

8:30- 9:00 - Coffee and Danish

9:00-11:15 - (Break from 10:00-10:15)
Speaker - Dr. Edgar Sibley . . . . p. 49
Topic: "Are DBMS Objectives Achievable?" ← e.d

11:15-11:45 - Speaker - Willis Greenstreet - Director - Management . . . p. 84
Information and Data Systems Division

SPEAKERS

M. H. Schwartz

Mr. Schwartz, now retired, is the former Director of the Division of Management Information and Telecommunication at ERDA. He has many years of experience in training government managers in the use of database management and information systems.

James D. Powell

Dr. Powell is an Associate Professor of Computer Science at North Carolina State University. He teaches courses on Database Systems and participates in research in this area. He is a consultant on database concepts to several companies and has been the principal investigator in the development of several information systems.

Leo J. Cohen

Mr. Cohen is president of Performance Development Co., and has written several books on operating systems, DBMS and performance measurement. He is also the editor of Database Journal and is a national ACM lecturer.

Edgar H. Sibley

Dr. Sibley is Professor and Acting Chairman of the Department of Information Systems Management at the University of Maryland. He is a consultant for several corporations and government agencies in the use and implementation of DBMS packages. He is a member of ANSI/SPARC ad hoc working group on DBMS, and has published and presented many papers in the field of DBMS.

NATIONAL COMPUTER CENTER

The National Computer Center (NCC), under the guidance of Director Sam Brown, provides programs, and operates computers and automatic data processing (ADP) systems for the Environmental Protection Agency (EPA), government contractors, and other government activities. NCC provides technical advice and assistance in these areas and conducts independent research and exploratory development.

NCC, as the prime EPA computer resource, is the Agency technical focal point in such areas as Data Communications, Database Management, Software Research, and Data Center Management.

DEPARTMENT OF COMPUTER SCIENCE AT NCSU

The Department of Computer Science at North Carolina State University has well rounded graduate and undergraduate programs with emphasis in computer systems, languages, operations research, analysis and synthesis of digital computers, numerical analysis and information systems analysis and design.

Mr. Leo J. Cohen
Performance Development Corporation
Princeton, New Jersey 08540

Good morning, everybody. Let's start off with questions. The question is "Name two adjoining rooms in the Warsaw-Hilton." The answer is "Database." The question is "What is the question?" The question is "What do we need database for?" Database does one thing very well and its been very carefully documented. One thing you can say for Database, Database Systems, Database Technology, Database Management: A lot of people are gainfully employed. The theme of my discussion this morning is "What is the question?" As a consequence of knowing what the question is, what kind of organizational attitudes (I can't say anything about organizational structure) are necessary to achieve a database answer to that question? That is the essential issue in the business of developing database for an organization big or small. Namely, why are we trying to do it? What requirements do we have to satisfy to consider database systems?

I would like to say at the outset that the technology of database and database systems is well established. It is complex technology, but by no means is it the most fantastic thing that anybody has ever conceived. IBM operating systems are; but database is a well understood technology. We know a lot about it. We understand the technology reasonably well. Given a certain set of conditions that have to be satisfied technically by the database system environment, there are reasonably well-defined methodologies for achieving an architecture for the database implementation.

What is not so well known is what causes the many failures (some of them roaring failures) in the database field or in data processing where we attempt to use database as a tool for development. It is the lack of a managerial commitment, the lack of upper management understanding and certainly the lack of any notion of why we're building the database system in the first place. Now, I go in and out of many companies. My company is in the consulting business and in the six years or so that we have been doing database system consulting, we've probably done about 350 database system projects. In that sense, I claim that I know whereof I speak. Our efforts are not so much aimed at solving explicit technical problems - it is almost a by-product as providing some organizational thrust towards stating what problem has to be solved. We go through a process. It's called a service analysis and we have refined the technique over a long period of time.

Service analysis is very easy to do. However, getting it done is the real problem. Bridges are easy to build. The technology of building a bridge is well known and straight forward. You get a spot, and you get some stuff and you get a lot of people and you build it. You hammer a nail and by the time you are all done, the bridge is up. However, have you ever considered the issue of getting all the pieces and parts that go into making the bridge including the people to the right places, in the right sequence at the right time? That's an immense problem. Well, if a bridge is complicated, how about a chemical plant?

One of our clients is the N. W. Kellogg Company. They build chemical plants and they'll tell you that building a chemical plant is not complicated. That they know how to do. They've got all sorts of talent, the high pressure lines, the high temperature lines, the electrical lines, etc. The technology is very clear. Their major problem and a problem on which they are spending close to seven million dollars to satisfy in data processing terms, is how you get the things ordered, delivered, stored, introduced to the construction and constructed in the right sequence at the right time to produce a project on schedule. That's a tremendous problem, and they have done it heretofore within very rigid way verging on the military-management organization. When I say verging on the military, I mean there's a person who is called the project manager. With the building of a *plant that person has all the clout including a flag on the left* front fender of his car. He's the general of the army, I can tell you. If he slams the door, everybody takes notes. I've watched that happen. In that sense, the issue is what has to be done in a database system environment in order to produce an understanding of why we are building that database system environment and, as a consequence, what are the organizational requirements in order to carry that off? I'm not going to tell you how to structure EPA. That may be a mythical issue. I can tell you, however, the roles that have to be played and it's your job to determine how to best arrange yourselves in order to play those roles.

Organization structure has one overriding characteristic. The structure must be able to deliver information up the structure to the decision-making levels, wherever they occur, and the decision-making levels must be able to deliver the decisions and the enforcement of the policies that they imply down to these information-gathering levels. That's the whole meaning of an organizational chart in the first place and of organization in the second. The formation and implementation of those decisions depends on the ability to accurately gather and produce information which depends on the organizational structure. In a database system problem environment, I want to be able to do exactly the same thing and therefore, not only am I going to talk about a service analysis, but I'm going to ask questions about the structure of the organization that we are examining for database.

I've got two levels of structure. The first level of structure is in the title of my talk. Namely, how should the EPA be organized, or as I have said, what organizational roles must EPA carry out in order to successfully realize database. Secondly, we've got to understand what we are trying to do with database and I'm saying that means understanding the organizations that we are dealing with. The second one is much easier than the first. I can't tell you how to begin to organize the EPA to satisfy the development of database. But I can tell you how to observe and analyze the organizational structure of the EPA in order to develop a successful database system project.

Before I get underway, let me give you some definition of terms. First, let me describe the difference between a database system and a file-oriented system. I'm going to be using some terms as I do that. I'll define those terms with a little more accuracy and then perhaps we can draw from that some definition of what a database system is. I don't have a smooth definition of what a database system is like. It's many things, at many times, to many different people. It depends on who is selling what, but I'll make an attempt.

Let me start it off by saying a word or two about file-oriented systems, some of the problems that they have produced, and therefore the reason why database system technology is of interest. I'm going to make a very key point in this. Namely, the database system technology does not solve these problems. It provides us with a tool to solve these problems. It gives us a potential for their solution. By no means, by its presence, does it solve the problems. I want to make a very strong point on that subject when we get to that part of the discussion. It's something that the buyer has to beware of, and therefore I want to caution you.

Let me make some simple points. In a problem-program file relationship, a verbal relationship is what characterizes our file-oriented system. The boss comes in and says "everybody gather 'round". "We're going to have a staff meeting because last night I had a dream and the dream is called 'automated inventory control'." The staff stands up and applaudes. One of the most brilliant ideas the organization has had for months and he says "what can we do to achieve automated inventory control" and somebody says, "chief, I've got an idea." "We'll use the computer!" and he says, "that's a great idea." "Now that we have got the requirements laid down, let's get right into the job." So everybody runs home and starts to program. That's the way its done. A couple of days later, the programs are all written. They get machine time one night; they are all debugged, the system is delivered, and everybody sits down and starts doing a flow chart and the documentation. That's what takes so long, by the way. Systems get delivered in a couple days. It's the documentation that everybody insists on that takes two or three years. But we all know that.

The point is that the problem implies a set of programs which, when they are executed, will deliver a solution to the problem. We implement some programs and then run these programs and presumably they'll produce the results that satisfy that requirement. This program will have to operate on some data so we associate the program with some files of data and immediately we have that word association. The program and the file are intimately connected! 'Intimately' means that the program encodes in its logical or algorithmic structure information about the shape of the record, the kind of data that is in it and probably the sequence in which the fields occur, etc. Now suppose this data architecture is modified, e.g., shuffle the fields around the record. Suppose I originally had them sorted by part number and changed the sorting order to sorting by bin number and geographic location in the warehouse. All the programs that used to operate assuming part number order will now be wrong. If I change the part number from 15 bytes to 28 bytes, then, unless I change the programs, they will be wrong. So we could say that if I have a collection of programs which operate on this set of data and I modify the data, then in all probability, some of those programs will have to be modified because there is a strong degree of program-data dependency. It's not the worst thing that could happen. Your pay checks probably come from the file-oriented system somewhere. At PDC, we produce our pay checks from a micro computer which is, in fact, a file-oriented system. After all, processing thirteen paychecks doesn't take an awful lot. However, then is a problem with a data and file-oriented system.

I don't mean to imply that this program data dependence is bad. Nobody else may care about the data so you can make a few changes and keep a bunch of maintenance programmers busy. Nothing wrong with that. They could be out in the street stealing cars. However, if I wanted to use some of this data for another purpose and I abstracted that data from this set of files and perhaps rearranged its format, its not the same data anymore. Now it may be called the same information but its not quite the same data and I must write new programs which process the newly created files. I now have two different file-oriented systems with some common data components. If I change the data in one of those file systems, then in order to maintain data integrity, I'm obligated to change the data in the other file system. You at EPA and every other government agency and 98% of all the organizations that use data processing face that problem. You become aware of what major and ancillary updates must be run in order to keep common data concurrent. Now, the problem becomes serious when there are eleven different copies of the same data in eleven different systems. It becomes serious because its very easy to forget or because I can't do all the updating all at once or because the cycles for updating are different in the different systems. I update something once a week, something else every two hours, something else once a year. But I am forced to make these cycled modifications because common data is being changed and sometimes it doesn't get done or it doesn't get done quick enough. You end up with outputs which depend on essentially the same data but which are different because there is no current version of the same data.

Now, database offers us a potential for solving such problems. It does not guarantee that they will get solved. It only offers the potential. The resolution of that potential depends on the definition of the problem A and then its solution B. In the database system problem environment, instead of defining a set of programs whose execution would provide a solution to a problem, I am more interested in the data that must be present somewhere in the database if that problem is going to have a solution.

Take the payroll problem as an example. In order to satisfy the needs of the payroll problem for a particular person, I must have the person's name; I have to have the person's hours, and wage, and all the other items that go into making out somebody's payroll. I don't care how its organized. I don't care what the formats are, what the arrangements are or what the relationships are. None of that means anything at the outset. I'm interested in what data I need in order to resolve a particular problem. Now accept for the moment that given a database with the suitable architecture, I can, under any circumstances of inquiry, find any data that's available up to the control on privacy. That means I can always write programs which will resolve the particular problem. It means I can always write programs but with greater or lesser degrees of performance efficiency.

Now it depends on the talent that is available, what you want to spend, and the system resources that you are willing to commit; but if the database has a suitable architecture, and contains the appropriate data, I can write an applications program which will resolve the problem. It follows that another characteristic of database is that I should be able to define new problems that involve the data that's already in the database. There is a crucial point. If I then add some needed data to the database, I obtain an augmented database with the proper architecture that allows me to write a new program to implement a solution to the new problem. I've produced a flexibility that is very valuable. I've made the data the essential resource of data processing development, not the program. That's another crucial point.

You discover in many organizations that in each class of problems or problem areas, there is a central set of data which is associated with all those problem areas. If you are a bank, the central set of data is the names and the addresses of your depositors. If you are an insurance company, it's policy numbers, names of the insured, the ages, etc. Now, the distinction between the file-oriented and the database system is that in the file-oriented system, the data is a resource for a particular problem. In the database system, the data can become a resource for the whole problem environment. This leads to the following first observation about the responsibility of management in organizational structure in managing the development of the database system project.

If there is a natural starting place in the development place, then there is a database which supports the start. Let's call this Phase A, Stage A of the database in the data processing development effort. Now, there is usually some part of the Stage A data which is usable for Stage B and I'll add some new data to the database in order to complete the data requirement for Stage B. Then there is Stage C and I hope to use a lot of data that is already in the database and add some for Stage C, etc. I hope to reach a position finally where the new products that I'm producing out of my database - data processing will, for the most part, find all of the data that they need already in the database. That's a resource. Then you are ahead of the game.

We work with an outfit in Houston which is interested in being able to account for the effective pollutants caused by drilling in the Gulf. Let me assure you that what I know about this problem is restricted to the database management package they use. I'm not even sure of a lot of the verbiage involved even though I've been around this a long time. But their interest is in collecting data associated with pollutants in the water and the structure of the underbay and the drilling and the drill locations, etc. Over a period of time they have produced an extensive volume of data that characterizes everything that goes on in the ecological sense in some big piece of the Gulf. Now, their problem is to be able to take profiles of this data to make sense out of it in a particular context. Well, nobody sat down and collected this several millions of pieces of data all in one guess. It was developed over some significant period of time, level by level, stage by stage, problem by problem, leading to a quality database resource that supports the evaluation and the investigation of such problems. That's the objective.

Now imagine EPA supporting a bowling league. Let's suppose that 35% of everybody that works for EPA in Washington is in a bowling league - an EPA sponsored bowling league. One of the members who is a bowling analyst attends a seminar and finds out everything there is to know about database. So on Tuesday night, he sits down and puts together the bowling league database system BLDBS. As a choice between writing a paper on it or debugging it, he decides that since the match is on Thursday, he's going to debug it on Tuesday. He completes it, borrows one of the microwave transmitting inventory recording terminals from Defense Supply Agency, (his brother-in-law works for them) and takes it into the matches. Rather than using a grease pencil, they sit it right down there on the table and all of the scores are recorded. What do you need for a bowling league database management system? You need the name and addresses of bowlers. You need their home telephone numbers. You need their job location, their job phone number, the name of the team they are on, the name of the league they are in, their handicap and you have to be able to record scores as they come. That is all you need, and you are all set to go and you sit there pecking away, recording everything as it goes along, taking time out to roll your frame. You can do a lot of things with a system like that. You can have real time, online, strike management recording! That way you will be able to tell which team owes what how many beers - a very important outcome of the entire evening. Now I can imagine a personnel director who is also in the bowling league saying, "Tell me all you know about EPA non-bowlers!" All I have to do is have the names, addresses, the home telephone numbers, job locations and telephone numbers of all the non-bowlers in EPA. Then, I will put in their job descriptions, GS classification, etc. into the BLDBS and what I will end up with is a personnel database. We started out with a set of data which was useful to the next things we developed.

The concept is typical of successful database development and it implies one very important thing: that we have some ides of what it is we are trying to build. It absolutely depends on organizational responsibilities. Let me assure you that if you say to the data processing department, "Jack, I want you to put a couple of programmers on it - get them started on something and write a few programs and let's see how it works." It won't. I guarantee that. By the way, that sort of thing keeps PDC, my company, and others in the consulting business very busy throughout the country and certainly helps our gross national product. It won't work. Programmers will see only their piece of the problem. They will build and design a database and use a package to implement a database system for their piece of the problem. Somebody else will do it for their piece of the problem and what you'll end up with two pieces of the problem. They'll look like file-oriented systems but they'll have the major disadvantage that instead of using operating systems access methods to data, it will be using database management systems access methods to data which, for that purpose, are terrible.

As an access method, I can safely say that database management systems are a real loser. They work when they operate on common set of data for all users concurrently. That's what they are designed for. That's what they do best. That's a tough message to get across. I don't know why I have been saying things like that for a long time and it's still necessary to say it. It doesn't bother me to have to say it. I think it is a fundamental attitude issue. If you're concerned with structuring EPA so that you can have an effective development of database data processing, you must consider it a managerial responsibility for understanding what the objective is and breaking that objective into a stage development such that the database becomes a useful resource. That's the role that has to be played.

I want to start from there and develop some of the details of playing that role. Now, in that discussion, I have used some terminology. I'll take a few minutes to expand on that terminology.

I've used three terms. It is essential to understanding differences in these three terms. It's very easy to mix them up. First, let's take database. I want you to understand the notion of database as an absolutely passive element in your data processing environment. The database does nothing on its own. If it does anything on its own, you're in a lot of trouble. There's something funny going on in the computer. The only thing you can say about a database, in absence of any other marks, is that it will hold down a building in a high wind. It doesn't do anything. The data is the passive resource.

I have also used the term database system. By the database system, I mean how this data is arranged, organized, and given on architecture, to suit the needs of the problems to be solved. I mean an arrangement of architecture that creates the minimization of redundancy and a maximization of relatability for the problem environment. I want to make a remark on that - a very simple straight forward remark. There used to be (there no longer is) a database management system on the market called the Time Shared Data Management System - TDMS - and it was made available by Systems Development Corporation. I know of at least one place it was used. There were probably two or three. It had the property that for any given set of data, it made access to any field of the database direct and automatic. The problem was that it needed one entire 360-65 and it carried its own operating system. In other words, you didn't use the computer for anything else but this. Well, in a sense we could say that TDMS was a simplified brut force approach to the resolution of the database system problem. Forget the requirement, just give them everything! I suspect that the reason why it is not available today is because nobody wanted to devote a full computer to just that particular problem. It's a guess. I know the system specifically because I was able to sit down and use it at the Los Angeles Police Department.

I didn't do any good. I was convicted anyhow.

The third term is the database management system package. Now, that's very much different from the first two. I want to emphasize that the DBMS is nothing more than a tool for describing the database to the computer and manipulating the data in it under command. It's strictly a tool for that purpose. Unfortunately, they are not sold as tools; they are sold as "databases or database systems", the implication being that if you buy this package I have under my arm, sir, you don't even have to mount it on your computer; just sit it at the console. It will automatically create a database for you for only $42,500 plus ten percent down and a dollar a week until judgement day - and it includes maintenance. Now, there is nothing further from the truth.

When I was a kid about 15, in Philadelphia, I belonged to a club called the Yellow Jacket Speedway Juniors. The Yellow Jacket Speedway was a dirt track in Philadelphia. It's long since been turned into a parking lot; but the Yellow Jacket Speedway Juniors used to build three-quarter midgets and race them. A three-quarter midget was a go-cart with a lawn mower motor in it. I used to race them around Yellow Jacket Speedway. I was really into that stuff. I was too big to get into a three-quarter midget because I was this size when I was 14 or 15 but I used to build it and I really loved it. That was a big experience for me. In the summertime, I worked at a machine shop to learn how to build all these fantastic things that I was trying to put together with nuts and bolts and what have you. I learned how to run machine tools, how to operate a drill press, an engine lathe, turret lathe, etc. I really got good at those things. I could take them apart and put them together. I could maintain them. I could dress bits and even set jobs up. I discovered that I knew a lot about a tool but didn't really know anything at all about how to design something for serving some purpose. I could build someone elses' design with the tool.

Well, I'm saying that we've got the problem of defining what should be in the database. That's a requirements problem - finding a technical solution that will then give us a database system and using a particular DBMS to create that solution or that tool. It's not a programming language. That's a mistake. It's not an access method. That's a bigger mistake. It is a tool for implementing the designs of our database systems. So, the database is the passive data resource. The database management system operates on that resource under command and then the application programs that are database oriented, the so called query programs, which add activities to the DBMS relative to that data and produces the results. I will jointly refer to all of these three things as the 'database system'.

Now, I have a definition of a database system: it is the query programs designed to drive a database management system package operating on a structured set of data designed to support the solution to our problem environment. It's not a satisfying definition, because it isn't neat. It's not concise. It is very functional. We draw the line around the query programs making reference.

I'll give you a simple example, an everyday example. Consider an encyclopedia. That's clearly a database. In fact, it's a database with a specific key technique. Namely, every article has got a title and in that title, somebody selected a key word and used that key word as the basis for sorting all of these articles into a sequence in the encyclopedia. Now, if you want to find the date of the last Irish potato famine, you would search the books until you found the book with the I's in it and go alphabetically through that book until you found 'Ireland'. Then read the article and see if there is anything about potatos and famine and find the date. Doing it that way saves you from going through the entire encyclopedia looking for that information.

Further on back is the index. We look up 'Ireland' to get a set of pages in which Ireland is discussed. Then you look up 'potatos'. You get another set of pages in which potatos is discussed. And you look up 'famine' and you get a third set of pages. The common page numbers among those three set of pages describe where Ireland, potatos, and famine are discussed. There you will find the date and that, by the way, is a fully inverted file organization structure. That's the very classical example of file inversion. By the way, it really works with an encyclopedia. It's a great way to locate anything in an encyclopedia with content and addressability.

Now, take my encyclopedia and cut all the topics out, page by page, throw them into a great big barrel and swish them around. I have the same database as I had before. If I cut carefully, every article that was in the old one is is this one. The problem is in the database system. I've lost the system part of it. That's what is wrong. Now, if somebody asks me to find the date of the Irish Potato famine, it may take a little while - months. That's the difference between database and the database system.

I think that it is important to recognize the responsibilities of the organization with respect not only to the database and what goes into it, but the sequence in which things go into it. Those are technical issues that demand organizational responsibility. There is no way around that. If somebody doesn't direct it, if somebody doesn't decide, if somebody doesn't take the responsibility for insuring that the requirement statement and then requirement analysis is completed, you will end up with a disaster and a lot of embarrassment. I can guarantee you that!

Let me make the point that database is not the answer to every data processing problem. I think that it is legitimate to spend a couple of moments discussing that. You may have a system in which all users are getting accurate and timely results. That, by the way, is not the usual thing; otherwise, people wouldn't need to use database, wouldn't need consultants, speeches at every conference, etc. They wouldn't be worried about the quality of the data processing they're producing. It may, indeed, be the case that you've got good results and to change good results because of the database is ridiculous. I see absolutely no point in it. It's too expensive to say, "Well we got tired of using our old system. It worked perfectly and everybody loved it; we're just sort of bored with it. The programmers are demanding some challenge." If you can afford that kind of expense to keep your programmers happy, you're running a disneyland out of a data processing organization. It's ridiculous to trade old systems for new systems before the old systems even heat up. You may have a lack of technical expertise.

In Washington, you are located where there is a lot of technical expertise, but there are many organizations that are not near convenient technical expertise. An example is the International Nickel Company who asked us to review their DBMS package selection. If you mine and smelt nickel ore, when it rains it comes down as sulfuric acid. So, for about fifty miles around Copper Cliffs, ain't nothin' growing. It's so bad, that's where NASA sent the moon astronauts. They camped out there one fall before the snows come. It was cold enough to freeze the ground and it looks just like a moonscape. They camped out there for three weeks, practicing being bored.

When we went up, they asked us if they should consider going DBMS. It certainly was directly applicable to the class of problems they were trying to solve except we didn't notice any hotshot database technicians standing around street corners looking for jobs. In fact, in Copper Cliffs, Canada, the International Nickel Company gets all of its programmers by teaching them FORTRAN in high school hoping they'll hang around for fourteen years. They even send some of the better ones off to Toronto for a college education on the deal that they'll come back for four years. They have a lot of very young people in the data processing department because nobody stays there longer than the four years. However, we recommend DBMS for the simple reason that even though they didn't have expertise in database administration, management was committed to the support of the database system project. Well, there is a good example getting to a database project without the necessary technical expertise available to do it.

You have to be very careful that, in your own organization, the expertise is available to do the database system project. There may be a lot of very smart people that are busy building the current systems and don't have time for database. The answer may be to hire another twenty-five COBOL programmers. I'm not saying that the technology is all you need it - I'm saying it is different. It will cost a lot of energy, a lot of effort, and a lot of money and time training people in the database technique. It's not obvious. It has to be accounted for and that's something management is responsible for and must be aware of.

Dr. James D. Powell
Department of Computer Science
North Carolina State University
Raleigh, N. C. 27650

My task is to present case studies involving database management systems. In preparing for this presentation, I had to ask myself why one would want to deal with case studies? The main purpose is not necessarily to go through and look at the details of the particular systems. The details themselves are not as important as what we can gain from the experiences that others have had - what are the activities that they were involved in that we want to avoid, and what are the techniques that they have used that we would like to build on in the future. Taking this framework as a foundation, we will consider at least two case studies - two examples where database management systems have been used. One of the examples was unsuccessful and one was successful. These examples are in organizations that have a structure that is similar to EPA's structure, at least relative to computing. The organizations have independent entities that share and use a central computing facility. In both cases, a database management system existed in the environment and the individuals wanted to use this particular system.

Now, what I would like to do first is to give some background information on the first problem that we will address. This particular organization had a series of field offices that were scattered in about eighty different locations. The locations were fairly distant from the central office and from the central computing facility. The field offices were involved in collecting information. In actuality a preprinted form was available in each field for individuals to fill out and transmit to the central office. These forms related to the collection of information on a particular sample or particular group of samples.

The information on a particular sample was collected periodically. Information was included that changed values for a particular sample, that added new samples to the database, or that added new detailed data about one of the samples. The data was transmitted to the central office and keypunched. It was put through an edit run and processed against the master file to update the information that was on the master file. There were approximately fifteen hundred to two thousand transactions per day. By a transaction I do not mean a new sample was created; but that additional information was obtained on a sample or there was a correction of some of the information relative to a sample. Each one of these, either the correction or the new information would be classified as a transaction. There were approximately five hundred thousand active samples and that number grew at the rate of about thirty thousand samples per year. In this particular environment, the paperwork of just handling the changes, the modifications and additions was staggering. In order to make sure the data was correct, a loop was set up. When the data was edited and run against the master file, a printout was made and the information was sent back to the local office. At this point a verification was made to insure that the proper data was received and put on the data file. If not, a correction was sent back.

Now, the data was used to produce a series of bi-monthly reports for the field offices. The reports were made on each individual sample. The reports collected the data and made an analysis of the information. Basically, twice a month, one had the cycle of collecting data, producing a series of reports and sending them back to the local office. A decision was made that this process was involving an awful lot of man hours. They also needed to have a better interaction with the field offices. They wanted to be able to process information in a more timely fashion. The administration of this particular department made a decision independent of other considerations that what they were going to do was to take the first portion of the process, the input mechanism and put it online. This change would allow people in the field to query the information, to update it from the field offices, and to verify that the data was correct. The portion of the system that did the analysis and produced the bi-monthly reports would be left as is.

How did they decide to make these changes? They knew that a database management system existed at the computer facility. The database management system was a system that could update very easily. It was a system that could query very easily. So, what they decided to do was to use this database management system in conjunction with a communication processor to set up an online system. The administrator called his computing staff to a meeting. He also called in some representatives from an outside consulting firm. He described his decision and asked them to give him an estimate of how long it would take them to implement the system. Now, he did indicate that, in his mind, the major objectives that were to be accomplished by this system were: one, to eliminate the initial paperwork, two, to provide a more timely service to the people in the field, and three, to increase the accuracy of the data. These were his goals and what he wanted accomplished. The inhouse data processing manager has a solid team of programmers and analysts. But he had recently lost, to outside contractors, one of the major projects that he needed to keep his staff busy and show that they were productive. In a sense, the discussions led to a bargaining between the outside contractor and the inhouse computing group as to who could get the job done the fastest. It was finally agreed that the inhouse programming staff would take on the task. They agreed with the administrator to have the job done in approximately four months. They felt that this estimate was reasonable, because with the database management system, they had a tool that they could use that would allow them to eliminate a lot of the programming. For many of the activities that were required the database management system would be a valuable tool and they would be able to move along smoothly. When the agreement was reached the development started immediately. Maybe, I shouldn't emphasize it, but what I'm describing to you is an activity that actually happened. The database system was a very complex database management system. It is, in fact, one of the most complex database management systems on the market. The system is designed to handle extremely large volumes of data. It is a very flexible database management system. It can give you a lot of power, but you have to know what you are doing in order to get the information out of it.

There were three basic components to the development. One, was the development of the software. One was the procurement of the hardware. They didn't have the terminals out in the field. And the third was the training of the personnel to use the system. Let's go back up to the actual software development. Even though the database management system existed on the central computer, the staff in this department had no previous experience with the system. In fact, they had read books about the system but that was about it. They understood its flavor and felt they understood what it could do. They decided that they needed some formal training. They got several days of training through the vendor's coursework. They also hired some consulting programmers with experience with the database system to work with them and design the main portion of the program. They laid out a pert chart to give them some feelings of what had to be finished by what stage in development.

After they got into the project, they began to see that things were not going to move along quite as quickly as they had hoped. They made a decision that they would split the development into two parts, inserts and modifications. The insert portion was given priority.

In addition to the software development, they also wrote very rapidly the appropriate procurement to get the terminals for use in the field. They purchased approximately eighty terminals, put them out in the field, checked them out, and as will happen everything went wrong. The modems, for example, didn't work and had to be sent back to the company. I don't think that is the level that we want to worry about now, but they did have a series of problems with the terminals. In the three and one-half months with, and this is the estimate that I was given, approximately three thousand hours of overtime, they were able to get the insertion portion up, running, and checked out. It was completed at four o'clock a.m. on the morning of the first training session. The training session went for approximately two weeks. The people learned, very carefully, how to use the terminals, what to expect from the system, and how to use the system. The instructors tried not to get their expectations up too high.

The system, as planned, would be very nice for the field workers. It gave them an opportunity to enter the data as they collected it. It also gave them an opportunity to query the database and make sure that that data is correct. They were, in essence, not hard to convince that this system was something that they needed. There were a couple of offices that said they didn't want the terminals. They had to be aided in their acceptance of the terminals and how to use them; but, this was accomplished.

During that two weeks of training, they were able to get most of the system up and running. D-day came. They flipped the switch and said, "OK. We have got the system up and running. It's up to you to use it. Enter your data, and let's go." Well, the operators started using it. When the operators had been at the training session, they had experienced excellent turn around time, things went very fast. Within hours after the switch was flipped, they were up to five minute turn around. The people were entering information and it was over five minutes before they got any type of response back. They immediately brought the system down and began to check it out. One of the things that they discovered was that in setting up the communication lines they had overloaded certain lines. They had not allocated the lines based upon probable usage. They allocated them more on a geographical area. They had to go back and redistribute the lines. There are problems when you do that. There were approximately twenty-three phone companies involved with the system. Other problems they found related to the complexity of database system. You must be very careful about how you design the data structures. So, they sat down and reviewed their code and making the modifications that were necessary to tune the system. Another thing, that they discovered, was that their computer was not large enough to deal with the volume that they were dealing with, assuming a reasonable expectation for response time. In essence, they had to go underground and order more memory. A lot of money had been spent on the system. These expenditures included personnel, terminal costs, and training time. There was a question. Should they pull out now? It was decided to continue the development but to implement the system in stages. Instead of starting with all eighty terminals, they took a portion of them, worked with them very carefully and made sure that they worked. Using this technique, they were able to get all eighty terminals up and running. The response time seemed to be reasonable. The people were fairly happy with types of activities that they were able to perform. It looked like things were finally going to work out.

They said, OK, we've got the interactive system going. It is time to turn to the second part of the development. This part involved removing the data managed by the database management system and putting it back into the file structures that the current batch programs used. They selected a sample section to run through the batch system. They took the data out of the database, transformed it into the format of the regular files and ran it through the batch programs. They also took the data from the parallel system and ran it through the batch programs. (It was not mentioned in the previous discussion, but during the trial period all data was entered both through the terminal and by means of the paper forms. Until the online system was fully checked out, it was important to have a backup system.) A comparison of the output from the two runs indicated that the online system produced correct response in only thirty percent of the cases. They hadn't worried about this part before because they hadn't had time. At this point in time, they were receiving approximately thirty percent accuracy on their new system. Operators in the field were well satisfied with how the system went relative to entering the data and getting the data back out; but, they would not have been very happy if they saw the responses from the system. At that point in time, the system again went underground. They took sixty out of the eighty terminals back to the home office and began to try to find out what went wrong. It turned out that there were problems with the programs. Because of the rush to finish, the programs had not been thoroughly checked. When I was talking with them last week, they were waiting the decision as to whether they were going to totally abandon the project and go back to or let's say continue with the paperwork. It is estimated that, including manpower and hardware, approximately two million dollars were spent over this period of time.

Now, I would like to point out two major deficiencies that caused many of the problems that occurred. One, I'm going to label as project management. I will put in this category many of the decisions made by the advisor to the administrator. For example, the statement that the project could be completed in four months. Now, maybe, one wouldn't want to put that decision in project management; but I believe it belongs at that level. Second, is the level of education. I emphasize this area because in the scenario the lack of education is not specifically mentioned. However, unless you understand the capabilities of the database management system you will not be able to understand its limitation and the complexities involved in application development. I believe that education is very, very important.

I would now like to run through another scenario for a different organization with, however, basically the same organizational structure. Again, the general background information is that this particular company is involved in the coordination of services. These are services to individuals in the field. In this particular case, they are health services including medical services, immunizations and maternity care. The funds for these particular service delivery areas (e.g. maternity case, immunizations) come from many different sources. The company realized that they needed computers to keep up with the activities in each of these service areas. They needed statistics to justify the funding. The statistics included such items as: how many people take advantage of a particular service; where are the people located; what is the age range of the applicants and what are the racial breakdowns of the individuals who use these services. In order to provide these statistics, the company had a set of computer programs designed to keep track of the information. The information could then be used on reports to the funding agencies.

About three or four years ago they began to evaluate their situation and to discuss whether they needed to change their procedures. At that time, they were having problems pulling the data together. Most of the applications were aimed at producing information for the various agencies who supplied the funds. The files were organized around this goal. But, the management began to realize that the files contained a large volume of important data and they wanted to use the information in their planning. They wanted to use the information to determine, for example, where services ought to be located. If, for example, it turns out that there is a large number of maternity services in a particular area, then they needed to get more employees who could provide maternity services in that area. They could not currently provide this data because it was contained in several systems. Another problem they discovered was that the same person kept showing up in many of these systems. Much of the information on him was the same in each system; however, each system had its own set of information that was different from the other systems. They began to see that throughout their system, they had a great deal of data redundancy.

They put together a team of people whose job it was to analyze the data needs of the organization. They attempted to analyze these needs relative to the company's goals. They wanted to be able to pull together the information on individuals, instead of having an individual's data scattered across eight different files. They needed to be able to know if this individual is receiving service from a variety of sources. They reviewed the data that was involved in the current systems. They also went out and interviewed the local offices in order to determine the types of operational data that they needed and would be beneficial in operating the local office. They had discussions with various levels of management to determine the type of data that was needed to assist them in planning. The team put this information altogether and came up with a plan. This plan was put through review process. It was reviewed by the local offices and by various levels of administration. By the time the plan was approved, it had the complete support of higher management. Included in the plan were some organizational changes. For example, the database administrator function and its staff had to be created, designated, and responsibilities had to be defined for the group. It, also, included a rather detailed schedule of training that was to be required for all employees. Programmers went through two courses involving seven days of intensive instruction. The analysts were involved in three courses. In the beginning, the company decided that the data administration function could be handled by one individual because the plan would be implemented very slowly. They were not going to start big. One individual could handle the administration of the project. The database administrator went to the same courses as the analysts plus one additional course. Because of these courses the database administrator had a great deal of training in the particular database management system and how it could be used to accomplish the companies goals. At that point, they developed a five year plan. This plan not only provided for a staged development of applications, it also required a staged installation of the online terminals. Each stage was to be completely checked out before moving to the next stage. They took the plan and began work. They currently have a database with approximately sixty thousand individuals involving approximately four hundred and twenty thousand services. One objective they have had from the very beginning was that they wanted more than just the operational system which kept data on the various services. They also have exceptional reporting and planning built into the system. For example, they are currently concerned with the immunization of pre-school children. Periodically, the system will produce a report on students who have not received their immunizations. It is anticipated that this type of report will prove to be extremely valuable. Another type of report produced by the system provides information for local agencies. The system is designed to produce a report on the clients that are scheduled to visit each local office during the month. This report will assist in local planning. Neither of these reports were possible under the former system. The system is also beginning to produce planning reports needed by management. These reports assist them in making global decisions.

They are now considering an expansion of the current implementation. They want to introduce new applications and new data for other service programs that were not in this initial implementation. They are, also, getting feedback from the local level. The local offices would like more information. As it currently stands, the local office can get information on a particular individual, what services he is involved in, what he is scheduled to receive and various additional information. But, they cannot get online summary information. The local offices are beginning to indicate that this information is something that they need.

The users out in the field seem to be very happy with this system. The database management system used in this example is the same system used in the first example. The development was, however, approached in a totally different fashion. The users have been involved in the development and the understanding of the system. They have been involved to the level that they are willing to come back and say, hey, we like what we've got but can we get a few more capabilities in the system. I chose, as I say, these two particular systems because they have the same type of atmosphere, an atmosphere of a computing environment with a centralized computer. There is a big dicotamy between the two developments. The major differences involve the level of planning and education that was involved.

I, also, have a third situation that involves the same database management system. This company has been involved with the database system for over four years. They have approximately sixty-seven online terminals that are providing a wide variety of queries to the system. There are thirty-five databases, many of which are interrelated. Again, I could go through the scenario for its database development but would be very similar to the second example. The primary emphasis was on planning and education. I mention this system to emphasize that with proper planning a successful system is possible.

PANEL DISCUSSION

PANELISTS

Dr. M. H. Schwartz
Retired Former Director
Division of Management Information &
Telecommunication at ERDA

Dr. James D. Powell
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27650

Mr. Leo J. Cohen
Performance Development Corporation
Princeton, New Jersey 08540

Dr. Edgar H. Sibley
Department of Information Systems Management
University of Maryland
College Park, Maryland 20742

MODERATOR

Dr. David F. McAllister

Question: At what level of organizational complexity do you start thinking in terms of DBMS terminology and concepts?

Panel: If I understand the question that you are really asking, where in an organization should the first issues of getting involved in this whole thing called database be addressed. For example, should programmers start talking about it over coffee; and if it seems like a good idea, write a memo to the supervisor of the programming department? That's not a good idea.

Question: Are you talking in DBMS terms or do you have to wait until you get to something the size of EPA as a whole before you start worrying about something as complex and sophisticated as DBMS?

Panel: I would say that it's not the number of people in the organization and I will give you a superb counter example. I just discovered not very long ago in a visit to Company X in California that they have about thirteen hundred people in their employ here in the United States and yet they have needs for starting the database system. Database is an obvious technique for what they have to do. So, in terms of the size of the organization times the number of its people, the dollar value of its activities or projects probably is not indicative of much. If it works, I'd leave it alone unless you can really demonstrate that either you can make it work more or work better in the sense that you are getting more out of the resources that you have in a profit making company. I would first assess that, frankly. It's now the case that the day you open the doors on the database system project, you spend a hundred grand. You get a package from somewhere, it costs a hundred grand more or less. Day one and you have to say whether or not you can save a hundred grand on anything you do. Remember you cannot resell the package. It doesn't have any used market value.

Panel: I take exception a bit here. It doesn't have to be a hundred grand, but I think there is another question to address before you ever get to that. That is, "What are the benefits?" and that has already been said. But if you can demonstrate some benefits from it then you can get in cheaper than that. The only question is whether you want to go database. Now, you're into one of the differences between database systems and they come all the way from soup to nuts. DBMS is big and it's expensive. You can go to the other end of the scale where you have what I would rather call "file management systems" like Mark IV or to database like System 2000. You can get into these for thirty thousand if you don't want online operation, but as soon as you go to online operation, it's another thirty thousand and so on and so on. So, you can start for close to thirty thousand. Now, what are you using it for? Presumably you are going to define the intended use and determine whether they can really save you money. It becomes a question of the tradeoffs between the amount of people time and the amount of system development time, vs. the cost and its origin. In other words, where is the money coming from? Which pocket does it come out of? Here again, a government environment is obviously different from the conventional environment. You're not looking for the same type of effectiveness.

On the other hand, you might be able to spend thirty or fifty thousand dollars when you can't get another line which might make it worth your while to go to a database management system. What the database management system is now doing for you is cutting out somebody else having to do it. In other words, you're transferring pure money which is sometimes easy to get for a line which is often more difficult to get.

Panel: Could I ask you a question? If you do decide to go, will you have to use a facility that already has the database management and what computer will you be using?

Answer: Well, we could possibly tie in with an agency computer in Cincinnati.

Panel: Does that system already have a database management system? Could you tie it into that?

Answer: I suppose it does. I came here largely to learn what DBMS means and I'm not really sure if they have one or not.

Panel: The reason I ask is if you are tying into a facility that already has a database management system, then the initial investment of thirty thousand to a hundred thousand dollars is no longer there. Then, you may want to look at it from the point of view of using it as vehicle to cut down on a lot of the software development that you might have. For example, if, in a research environment, you are going to ask lots of the inverted file type questions, then you either have to design software to do that or you can take advantage of the existing system. Again, there are tradeoffs which must be considered, but I think you have to look at it from the point of view that there may be an existing database management system that you can use. Then, the comment that Cohen made this morning is cogent and that is, "don't use database management system as an access." If it's just going to do the same thing that OS can do for you, you're not gaining anything. In fact, you are loosing time. You need an application where you're gaining something through database management systems. It may be just programmer time and people time is very important.

Panel: I'd like to make a general statement about dollars here. I'm responding to a remark about the cost of packaging. I could argue with that thirty grand. However, it doesn't make any difference. That's the first cost. That's the inevitable cost. Then, there is at least one other second cost - to learn how to use it and what it does. There is probably a third cost if you are going to use it for anything. That's deciding what the problem is, getting the design together, doing the documentation, etc. That's all those things called data processing. The point is, you don't come out of this cheaply and the reason I make this remark is that significant moneys get spent and they should be worth it. Therefore, I would say that, at the outset you have to be able to say it's worth doing. That was really the point of my statement, not so much the dollars. I go into enough commerical organizations and watch them spend millions on their database system projects. They don't even bother to ask whether or not it was worth it. I sometimes wonder. As long as they are aiming some of it at me, I really don't mind. I think you ought to ask the question up front. Does it seem like it is worth it or will be worth it?

Question: Are database management systems limited as to the size of the database, i.e. can they handle fifty million, hundred million, two hundred million values?

Panel: I don't know that. There are a lot of mythical stories about packages which are good for big database and packages which are good for little databases.

Question: So they are infinite?

Panel: Yes.

Panel: As infinite as you are willing to spend.

Panel: Really the problem is that you can get to a certain point, but if you ever got to that point, you probably couldn't afford it anyway. How much are you going to have online, rotating, waiting for someone to ask the question? The answer is by the time you got a room full of a thousand disks, you'll probably be getting out of space.

Panel: You'll probably run out of electricity. I've heard it said many times that System 2000 is a fine package for small database systems and yet a life insurance company uses it for its life management file. It probably wouldn't fit into this room. I can certainly see it applied to some rather massive database system problems. They are all just as good for big and small problems.

Question: Are database management systems computer specific? Are they built for a certain make of computer and are not readily interchangeable?

Panel: Some vendors have made them interchangeable. System 2000, as an example, runs on 370's, UNIVAC, CDC, etc.

Panel: It's almost as easily transferable as COBOL.

Panel: You've got to make a distinction between what's transferable. "Transferable" is not the right word for what is "interchangeable". For instance, suppose you have System 2000, a set of application programs developed to operate on the System 2000 database and you wrote them in COBOL. Then the application programs have the degree of interchangeability (portability is a better term) that COBOL has. The DBMS has virtually no portability. That vendor has to have a comparable package for the other machine. In making a conversion or rewriting it for the other machine, it probably included something to take advantage of the features of the other machine; and your COBOL probably isn't going to work as well. It's called "sideways" or "crabwise" compatibility, I think. Then, there's the database and that's the biggest problem. It may be the same DBMS. It may be the same scheme which describes it, but it's going to be wrong for the storage device on the other computer. The architecture probably won't work well. That's a very generalized and generic response, but I tend to take the most pessimistic view of the possibility of transferability from vendor to vendor of anything except people.

Question: I understood that Systems 2000 is limited by the database - that it won't accept a very large one?

Panel: It depends on what you mean by "very large". There are limits on database systems always.

Panel: We have to put this in a slightly different way. There is probably no limit inherent in the package which is feasible to reach. For example, in an IMS physical hierarchy, there is a limit of two hundred and fifty-five segment types. If you ever get two hundred and fifty-five segment types, the probability of it running is zero. But that's an available size potential; and in the same way, you might say that there's some physical upper limit to things in System 2000. On the other hand, you'll run out of performability before you ever run out of the intrinsic capacity, whatever it is. For instance, several hundred repeating groups are possible in a given schema.

Panel: Fifteen hundred, but I can never think of more than three if I'm ever given an example. I've heard of people that have got to seven or eight.

Panel: What is being said is that the package offers not an infinite capability but it is certainly well beyond any pragmatic concerns. It's been my experience, by the way, that I've never said, "gee, we could solve this problem if only the package had more capability." That's never been the problem.

Panel: I'd like to return to that first question for a moment if I can because I have been thinking about it. The question was "what sense does it make to consider the use of the computer for the administrative purposes for a hundred-man laboratory?" Do you go from just plain computerization to the DBMS concept? You see, it depends on what you really mean by computerizing, what you mean by database management. Ed threw out a term, Mark IV. Mark IV is the name for a system that was developed by a company called Informatics in California. It's an accounting oriented system. It is not a database management in the sense in which we use the term today, although five or six years ago it was considered to be a database management system. The field has transcended the capabilities of Mark IV. Mark IV is, in fact, a very easy system to use. Many large organizations with very sophisticated kinds of systems around also use Mark IV. For example, the Atomic Energy Commission at its headquarters use Mark IV on classical data processing systems and after a year, some one hundred and twenty-five accountants and budget people, which was a big piece of the total AEC budget and accounting staff, were writing little Mark IV routines of all kinds, some to put up a little data system, mostly for the purposes of extracting ad hoc reports from on-going systems. Now, for a small laboratory, it is entirely possible that a Mark IV package would be the way to do the whole thing - to put up the system, to run the system, to augment the system, to query the system. Mark IV does not run as fast as a specific program addressed to a job, but the cost of the machine time might be very small. The savings and programming time that Mark IV represents can easily offset the additional machine time costs. As a matter of fact, machines are getting to be the cheapest element here. So, the answer to your question really depends on not database management systems, yes or no; but really "which one?" I think you would be very foolish in a hundred man organization to write specific programs for your specific jobs. I think that would turn out to be very expensive. I think you would want an easier to use facility. Something like Mark IV is a very attractive (I'm not in the business of selling Mark IV), but it is a very attractive option. So, this means that you should go away from here with an eye toward examining whether or not to use DBMS; but which one might be able to do the job. I know something about laboratory operations. I do know that if you have a laboratory like Livermore with thirty-three hundred people in it or some such large number, then automatic MIS is extremely powerful but there you have enormous sharing of all the costs. Now, the chemistry division is larger out there than your whole laboratory, but there are some scale factors here. To summarize all this, I would be very careful about sitting down and writing a bunch of COBOL programs to do this whole job.

Question: If he did tie into a computer system that had an existing database management system, are you implying that one ought to look first at the utilization of that before they write application programs?

Panel: No, not necessarily, because if the database management system happens to be IBM's IMS, you would not want to consider using that system; but if you found a Mark IV system somewhere in town which you could piggyback, well, you're going to have to piggyback somehow or other. You're certainly not going to go buy a computer for a hundred-man laboratory, at least not at the outset and certainly not for just MIS. If you had a machine available with Mark IV installed upon it, then you could piggyback.

Question: I would like Don to give us a rundown on what packages we've got at EPA.

Don Worley: We have two database management systems at EPA - one called System 2000 which is available on the system in Washington and the system in North Carolina. We also have DMS 1100 on the machine in North Carolina and an infinite number Mark IV type packages. I believe that Mark IV is on the IBM system, plus we are installing another package on the PDP 11-70's.

Question: Why hasn't anybody gone and talked to Joe Califano about DBMS being on the welfare program or does it not lend itself to that?

Panel: Do I just go and knock on his door and tell him you sent me? Are you sure I can get in? I'm sure I couldn't.

Panel: I'd like to answer that question. Let's get back to the point I made. Don't attempt to use an information system to solve HEW problems. They've got lots of real problems they've got to settle before they decide that the time has come to use DBMS. They've got structural problems.

Panel: I know we are sort of beating you over the head here, but I think the message I was trying to get across this morning is sort of portraited by HEW. Namely, the solution to their problem is not database nor data processing; their problems are strictly organizational. What we are saying this morning is that the database problem has its technical aspects and it could be a nifty answer to a properly stated question. Well, it's hardly the stated question of HEW. "We're in trouble" is not a stated question.

Question: Everything we have heard so far today has suggested that DBMS kind of locks you in for a fair amount of time. Maybe I would be looking at a three to five year span. What I am concerned about is how locked in do you get in respect to your hardware. Do you come up with a particular package on a given piece of hardware today and three years from now it looks like it is desirable to change hardware because that hardware doesn't work with that particular package? What's the cost going to be to convert? Is it as great as the original package?

Panel: Greater, because for the system you have, you have built procedures and operations around it that employ it. Because you will put all this stuff on a new machine. Even if the analysts leave it alone and simply reproduce it, the other machine will change it enough that there will be changes in the system around it.

Question: What if you are starting on it today with a staff with no DBMS experience?

Panel: You'll save the training in that particular area. That's part of it. Relative to the overall cost of a moderately size database system project that has lasted for three or four years, you're not talking about enough to sneeze at.

Question: Does that have to be a relative comparison? You're obviously going to be ongoing. You're either going to be developing in a traditional manner or a database manner. Are you going to be in any worse shape with a database system than with a traditional system at that point in time?

Panel: It's a question of what your strategy is. If your strategy is get in, make money, and get out again, what difference does it make? So many of the people that I worked with in industry are of the opinion that they're saving enough money while they are out, even if they are going to throw the whole thing away in six or seven years, are going to a database system. In other words, they have to assume a total scrap with no value except potentially the fact that they have got a well-documented system they are going to reproduce.

Don Worley: Let me inform the panel that this is a TV game show. One of the things you don't know is that EPA faces, in about three years, a major conversion and hardware.

Panel: Face the major conversion in money. There is a payoff if you buy a bus to conduct a profitable business. At the end of five years, you've got to buy another bus. A new bus (forget about inflation) will cost as much as the old bus, but it's a useful price to pay because of continual profitable operation. You're coming back to the issue of cost versus benefit. I think that is a central theme in a rational discussion of whether or not to get in the database system environment. It's got to pay off somewhere, somehow.

Panel: I think that this is something that will matter very much to the government environment. There's a suggestion that there will be changes to GSA rulings which will allow you to take conversion costs into account. This would suggest that if you don't want to convert to Brand X and you would rather like to stay with Brand A, you might be able to allow Brand A to bid a higher price and still make it because your system conversion is cheaper.

Panel: There is a technical potential for mitigating some of the pain in making such a conversion. Marty Zimmerman who heads up all computer installations for the Army (I think) is worried about sixty-two million dollars worth of computers.

They are all IBM machines and when they had to be replaced, they had to be replaced by an open bid. They can't go back to IBM and use something in a different color. Now, under those circumstances he's not too concerned about the sixty-two million dollars worth of replacement, but he is concerned about many hundreds of millions of dollars worth of software replacement. Two or three years ago they asked me the question, "What is the portability value of putting a database management system into a stand alone minicomputer that is appended to our computer or mainframe and does that database activity in what is known as a backend database processor?" This was the subject of a one day presentation I made at the NCC in Dallas and that's definitely a feasible technology. You get the potential degree of portability that is in the application programs themselves.

Panel: I would like to continue on two points. The first is the backend computer point. It's true that it is within the realms of technical possibility, but on the other hand, once again remember it was the bravest Christians that got the hungriest lions. The reason I'm saying this is that the civil service commission has just come through a procurement where they had a perfectly good statement of requirements for a backend computer and nobody bid. So, in other words, there wasn't anyone even willing to be a hungry lion on that particular one. I believe that civil service was lucky because they would be selling themselves down the road for failure for the next five years or for the next ten years while they were getting the bugs ironed out of the system which hadn't even been produced by any software manufacturer at the present time. The second point is standardization and I'm in favor of standards. Next week in Florida, NBS is running a symposium or workshop on the problem conversion costs, database management and standards. Now, the whole area of standardization is extremely important to the Federal Government. After all, there is no way that the Federal Government is going to keep on getting incompatible hardware and subsequent conversion costs; therefore, you are going to have to build plateaus on which we construct our system. Database management systems happen to be just such a plateau. Therefore, I'm in favor of standardization. I go as far as saying that even if we got the wrong standards, we would still be better off than having no standards at all. Now, that's not a particularly happy statement to make in front of most groups because they all say, "ah, we don't want to standardize the wrong thing." They've all got their own reasons for keeping themselves locked into that; but as far as the users are concerned, particularly in the Federal Government, I think it is absolutely imperative that we get standards.

Question: Don, you just mentioned traditional systems as opposed to DBMS systems. What do you think about the traditional systems.

Don Worley: I consider the Mark IV a traditional system, a file-oriented system. For our purposes, anything that is a database management system is usually a structured type of organization for describing data. A traditional system would use a standard file.

Question: Can you put together a number of different files to form a database?

Panel: With difficulty and sometimes not at all.

Panel: Not in Mark IV. Mark IV is an excellent example. You, in effect, append Mark IV to a file of your computer system. If you wanted to run on some other files to do some other things for some other people, you append to those other files. They have nothing to do with each other. There's no cross referencing of information or what have you.

Don Worley: I guess I will interject, again, to say that EPA has sort of used our own definition. We say STORET is a database and ARROS is a database because it sounded good. He gave us a new definition of database.

Question: What are the differences between the Chevette and Chevelle model DBMS's. Are they all basically the same?

Panel: That is a very difficult question to answer. I don't want to get too technically involved, but let me put it the following way. If you have a small database, then it's fairly obvious that you don't have the same problems as you do when you have a large database. For example, if you have an extremely large database, it's going to take you six or seven hours to load. You'd better not lose it and not be able to put it back up again. The problem of being able to offload it and reload it and to be able to maintain the system under adverse conditions are problems that generally have to be built into systems which have to handle large-scale databases. Now, part of the reasons you pay less for certain size of systems is that you get less of these goodies tacked on. The reason some of the panelists disagreed as to whether it was 100K or 30K was because when you go and talk to your local friendly DBMS dealer, he'll tell you what he's got, but if you want to have wheels on it, that will cost you another hundred bucks. If you would like to have something to steer it with, that will cost you another two hundred bucks and by the time you're through, if you're not careful, you're into a hundred thousand dollars just because you want all those basic features or you don't know which you want, so, you take them all. Hence, there is the problem of knowing how much you need. For example, most database systems start off by a person saying, "We need something and it would be awfully nice if we had it online as well." So, right away, you're into data communications. Now, a data communication package will cost you another fifteen thousand or twenty thousand dollars and so as you keep on adding on, it becomes more and more expensive. It becomes more and more complex. It's more and more likely to go down. You can buy a Chevy with all the accessories on it and it will cost you almost as much as a Cadillac, so why didn't you buy a Cadillac in the first place?

Panel: I want to be careful here. I would say that the following is true. There is no package which one can call the "best". I think that for a particular problem environment, type of data processing, available skill levels, organizational structure, lead times, etc., you can choose a best package for your organization. But I would go further in saying that for your organization, it suffices to close your eyes and take a package. You can always make it work and solve the database system problems in your organization. There would have been a happier choice if you would have taken time to be careful about the choice.

Panel: I have one friend who, tongue in cheek, says it's probably cheaper to pick one than to choose one. Pick one and use it, rather than choose one because it's going to cost you so much time and money to choose it.

Panel: It's too expensive to make the choice.

Question: To get to hear the rest of what I wanted to hear, are there logical additions? You mentioned data communications. What is the critical mass? What's the next thing you usually add on, the next thing to that, etc.?

Panel: Report writers, fast loaders, wire wheels, racoon-skin caps, long list. Take System 2000 as a good example. I guess it starts at thirty grand. If you get one of everything it can cost over a hundred and forty thousand.

Panel: That was what I was trying to say, you see. You can get a stripped-down version, but it certainly starts to add up. You decide which system you need and then you decide what features you need and then, in one sense, you decide what it is costing you because you have got things you don't need since some systems are monolithic. For example, with IMS there is a basic minimum you have to take. If you don't really need all the things that they are providing, you may be having to carry all sorts of core requirements and all sorts of extra storage requirements because of the stuff you don't really want, but that you can't eliminate. This is the point. IMS is a big system and most people who have used it find out that an IBM 360/65 is just about big enough to start using it provided you have an acre of core. It's a big system and you are going to have to use OS, well, in the old days OS MFT, or MVT with four regions. By the time you're through, your operating system is taking something like a hundred and fifty thousand bytes and then you're putting your database management system on top of that. In 1970, Chevrolet brought up a system on a 360/50 with a quarter of a meg of core and they didn't get anything done in background batch. Whenever IMS came up, it hogged the whole machine. Nothing else got through. Now, that's a pretty big machine to be just doing a few IMS jobs. I'm not saying IMS is bad. It sounds as if it is in some sense; but it's a big system doing a lot of big things sometimes. If you're doing small things, you don't need a bulldozer to break an egg. Well, you know you can break a lot of eggs that way very quickly if you want to. You pay your money and take your choice. You try to tailor the system to your needs.

Panel: You might mention that IBM is classic that way. In most of the systems that they design they try to be all things to all people. You have to read 27 manuals before you can use the system.

Panel: I hate and I love IBM at the same time. If you read the Data Pro Reports on systems, you'll find that most of the people who use IMS hate it. But find out what they are going to do about it. The answer is nothing because they are stuck on that particular system - because that is the only one that they'll be able to use or the only one that their management will allow them to use - because they are in an IBM environment and they are scared stiff to go to another system.

Question: I'm starting to get an understanding of database management systems and look back on the months and the years. What is driving us to database management systems versus the alternatives? I know we had a couple of examples, but they weren't too quantitative.

Panel: First, in a database environment, you've got to plan. If you looked at the two scenarios in Powell's talk, an implicit planning process was involved.

It forces you to make some solid decisions, to realize where you are going, and to move in those directions. It provides an organizational unity within the information process.

Second, suppose you have an inventory system and a file that keeps track of part ID, the amount you have on hand, etc. Assume you get that system up, running and working. Tomorrow, you come in and say "I would like to know, in addition to when an inventory is low, which suppliers are involved." So you set up a second file to handle suppliers which is fairly simple. Now, you say, "Let's put those two together and when I need to order a part, let me get the parts and the suppliers." Next you say, "I need some additional information about the suppliers, like their addresses and phone numbers." You set up a third file. Now suppose you are interested in a list of all the parts a given supplier supplies. That particular application uses the data you have available but it's in such a form that you have to do a lot of sorting and a lot of sequential processing or running repeatedly through the various files to get the particular data. So, if you start a regular file oriented application, pretty soon you can get your applications fairly intertwined where a lot of your programming is abstracting data from individual files in order to put it together in some form for your current application to work. Hence, the database management system helps to minimize complexity of programming in developing later applications. Assuming the same scenario, had I had a database management system at each stage I would not have to worry about the way the data is composed because the database management system will now provide the interface between you and the data. You don't have all the maintenance. So, by not having all this maintenance, you are cutting down on programmer time and you are cutting down on design time. You are able to get your applications implemented with less demands on your personnel.

There is another type problem. Suppose for example, I have suppliers in two files and a supplier changes his name. Now, I have got to worry about making sure that I don't have duplicate data. Now it's a matter of integrity within the data. It is a problem and hence a hidden cost in trying to maintain the integrity of the data. It is another benefit of a database management system: you can reduce the amount of data redundance in order for your system to perform. You will never eliminate it; but at least it's not in eight files. It may only be in two. Other provisions that are offered by database management system are recovery capabilities and security capabilities. These will probably be discussed tomorrow. I'll stop there.

Panel: I'll try to answer it from a lot of perspectives. First of all, it is always a matter of deciding what your benefits are. We have been saying this two or three times and I can't apologize for saying it because it needs to be said. Undoubtedly, you will save people time because writing programs versus writing a system within a database management system is generally something like thirty to forty percent cheaper. In other words, you are reducing the amount of code. You are reducing the amount of debugging time. You're likely to get your system up more rapidly. These are some definite advantages. Suppose you code the system in, say, COBOL and then you code the system in something like IMS or DMS 1100 or one of the programmable type systems. Then there is no doubt that you will cut down your costs and your debugging.

Another problem is susceptibility to fraud. You have generally five or six securities built in and you might very well say, "What is the DBMS doing for me?" It's all overhead and it's all going to cost me more money. But the answer is that we have been literally living on our short hairs in this profession. We've been assuming that we have gentlemen or ladies who are writing code. We begin to see crime creep in and all the problems associated with privacy considerations. If we don't assume that we have got to put those costs into our system, then we are fooling ourselves. So, compare the efficiency of a current system which has no privacy or no security considerations with a modern system which has got to have security considerations is like comparing apples and oranges. If you use a DBMS, you are generally getting a better system, you're getting better data checking, and you're getting the capability of recovery without a great deal of downtime and a great deal of problems associated with that recovery. It's all built into the system. It's not relying on the haphazard programming of your own programming staff and the fact that your own systems analysts have forgotten to do this or that thing which results in a lousy system.

In the 1969-70 era Chevrolet replaced a purely manual system with an automated system. In the manual system it took them up to seventeen weeks between the time an engineering change was made to the time that everybody within Chevrolet knew that that change had been made. During that time, procurement might very well have been procuring something based on the original value. The fact that they got the change notification down from something like fifteen weeks to six weeks was enough to pay for their entire system within a year. Now, that's a weird benefit, in some sense; but on the other hand, it is a true benefit because you are replacing an old system which was lousy by a new system. Maybe you don't know whether it was lousy or not; but it sure is better than it was.

Another point is the following: the system is up, you've got the data and now you can start to do something with that data, running whatever you like on top of that system as an additional benefit which you couldn't previously do because you didn't have that data up online. Maintenance is another point. Maintenance is more automatic with the database management system and it's being done coherently. Sweden went database for that very reason. They've been plotting a maintenance cost. They found their maintenance costs had gone up twenty to thirty percent and they were projecting somewhere about 1980 that wouldn't be doing anything on the data processing shop except maintaining! They saw database management as a methodology of taking that overloading of the programming staff and maintenance away.

Equitable Life has a very large system dealing with their insurance accounts. Their systems, in the old days, had lots of holes. They knew that their data quality was lousy but there was nothing that they could do about it. Then they put in systems which had good checking producers so that new data coming in was improved. As a matter of fact, they ran all data against their original file and rejected seventy percent of their file! Think of it! This is an insurance company rejecting seventy percent of their files because of errors that they can detect by machine. It's ridiculous, but it's true. I've heard of people in the army who've told me that they think they've got systems with ninety percent errors in their files. It's pretty sad to think that we are dealing with systems that are so lousy. Apparently we have been living on borrowed time.

And, don't kid yourself. The cost of recovery is really high. Equitable went through the following situation. They lost a file. It was a little error that came in and hammered one of their files and it happened to be on fully paid-up policies. They worked out that it was going to cost them five million dollars to get that file put together again and it would take them about three or four months. So, they called the actuaries in and the actuaries decided that the chances of somebody asking for their money on their fully paid-up policy was so small that they would be able to live with it. Hence, they never did recover that database. It just wasn't feasible as far as they were concerned.

Don't forget the crime issue. Someday there is going to be a big government administrator who is going to be on the firing line because of privacy. It hasn't happened yet, but it is only a few years away. What's going to happen? Jack Anderson or some other so and so is going to come in and he is going to find out that he can get into your database. He's going to publish it just for fun in the paper to show that he could. Then you're going to be up there with a rope around your neck. That's the day that everybody is going to start worrying about it and everything is going to start suddenly tightening up again. Again, we've been living on borrowed time.

Panel: Database management systems are much more than just a very neat file structure or a very neat data layout that facilitates access to it. The database management is really a comprehensive system. They contain many hardware-like and software capabilities and by going to the database management system, you buy the more ready availability and the lower cost of availability of lots of these additional capabilities.

Both of these gentlemen mentioned what was on my mind. Take the matter of computer security. Suppose you had a file that you wanted to have reload access availability to. Suppose that you wanted to have it so that some people could look at a certain part of the file and other people could look at all the file and perhaps other people could look at only some other part of the file. That's so expensive to do if you start from scratch and build an ad hoc system just to build that file and generate reports from that file. It's very expensive to add on the intricate software which will protect that file so that if the wrong guy tries to access it, the system will refuse to give him the information. That alone might be enough to say I'd better go to database management system. Not because I need to save the programming costs, not because I need to run a variety of applications across the same data, or not because the data stream cuts across many functions of the organization - simply because of certain other characteristics which happened to be embodied in these database management systems. So, now, to get back to your original question. It's a very challenging question you have asked. My response to you is that you should be very careful to have your mind open (forgive the motherhood) and be very careful that you are not overseduced. A little bit of seduction is a good thing.

Make sure you're not overseduced into bells and whistles that, in fact, that you don't really need.

Dr. Edgar Sibley
Department of Information Systems Management
University of Maryland
College Park, Maryland 20742

I shall split the talk into two major parts: The first will deal with the needs, methods, and advantages of Database Management; the second will deal with some common problems in its implementation in an organization.

The topic is database management systems, but the real topic is information systems, and information systems exist because there are users. So, therefore, it is worthwhile starting by identifying our users: their needs and aspirations. The first sort of users are "real" people. They are planners, managers, analysts, scientists. As you go down the line. you get less and less important to the organization; e.g., politicians, crooks, and even snoopers. You will notice that, so far, I have not talked about people who do anything with computing; to put it another way, one of the biggest problems with data systems is the fact that people forget that computers are there for people: "real" people, the users of the system. Presumably, these people are not very interested in knowing much about the internals of the machines. They are interested in what they can get out: therefore, identifying users identifies one of the design parameters. They only bother that the system is usable, and they have no real interest in topics like computer science or database management.

Now, the question immediately arises: How educable are these people? In other words, should we twist (i.e., "educate") these people so that they can use computer systems, or should we design computer systems so people can use them easily? The natural language people say "Give people a natural language interface, and they will be able to communicate with the computer." I say, give people a natural language and they will be able to make mistakes even faster. The trouble with natural language interfaces is that when you set up a natural language interface with other people (otherwise known as talking to them), you use a lot of redundancy in your discussion. Moreover, you may spend a great deal of time discussing what you are trying to do; therefore, in the end, you have resolved all the problems (you hope) before anyone starts doing something for you, or vice versa. Computers, unfortunately, do not have that sort of interface. In other words, if you have a computer, it is generally fussy. You get mad, but it still keeps on asking you to resolve a question that you think you stated pretty unambiguously. Of course, some people get mad at one another during their discussions, but they keep on talking. Context helps people. When you say: "I saw a man eating fish," are you talking about a shark or a person in a restaurant with a nice fish platter? These sorts of problems are resolved rapidly by people; but there are studies recently which show that non-mathematical but intelligent people, given natural language as an interface, often cannot sometimes describe exactly what procedure they wish to use! For a complicated question, we need the concept of "quantification"; for example: (i) Find people who have always worked in Los Angeles; (ii) Find all people who have worked in Los Angeles. What's the difference between those sorts of expressions? Obviously, they are different, but many non-computing people cannot tell which produces the answer to a well defined question like: "I want to know if we have someone in our firm who has only worked in Los Angeles."

Providing systems which are useful to people is our business and these people are generally not interested in the internals.

Now, there are a whole set of other people, "unreal people", if you like, or "computer people" who again appear in various categories that need to be serviced. These include: data gathering people, who collect the stuff we are going to use; database administrators who, presumably, from our discussion yesterday, are people who decide how to put things together, to package them well, to be able to make them useful, to be able to make them efficient in some sense for the users standpoint. We have: program managers, who are trying to make systems work for people within the confines of the policy laid down by the database administrators; application programmers, who write programs for people who do not want to learn languages; system and problem specifiers who are essentially the systems analysts; and the ad hoc queriests, who use a special high level language to pose a query. The ad hoc querist would probably use a natural language interface if it were provided! We have also operations clerks who presumably are often the real users of the total system; and the site operators who keep the system operational. Finally, we have systems programmers.

Now, why discuss all these people? Well, if we have a database management system and we do not make provision for interfaces which support them, then the system is useless or will soon be in trouble. As an example, if the site operators are not sure what is going on, then the next time the system crashes, you have got a disaster. If your systems programmers are allowed too much leeway and you forget that they are there, then you have potential troubles with penetration of the system, extra chances of crime, good chances of an expensive ripoff, or possibly they will just poke around in the database for fun (which is ok) but not exactly professional, and may violate privacy constraints.

If you do not have application's programmer interfaces, you cannot write large programs like payrolls, and so on. Presumably real systems are built to satisfy a community of users, and not all users.

We shall now look at the phases of operation of information systems: first, how do people get to know about the data that exists? There are really three types of mechanism for defining data: I call them the catalog, the dictionary, and the directory. These words are quite descriptive and have essentially their common meanings: catalogs are found in libraries; dictionaries are books which help with definitions and in finding out how to spell words; directories are volumes which help in finding a phone number or address. And these are exactly the sort of mechanisms that are needed in an information system.

When you are trying to get to information, you would like to know where it is. If it does not exist locally, you need to know how to get it: a catalog tells you about data not immediately available to you locally. I shall not talk a great deal about catalogs, but they are a good lead-in to the idea of dictionaries and directories. Dictionaries are books which tell you the normal way of writing or its meaning: they give the spelling and a definition so that you are in agreement with others. When you say "go", somebody else does not assume that you mean "come". This is a major problem in some organizations; e.g., for a large oil company, one part's "exports" are another divisions "imports"! Moreover, parts of the same organization may have totally different concepts for the same word. At our University we were recently faced with the question: "What is a fulltime equivalent student?" We were amazed to find that we have different definitions in different parts of the University; it changes from undergraduate to graduate and across professional colleges. We recently went through an agonizing period of about three months where we tried to define a few words at the top administration level. A high-level group tried to describe the major functions of the University and the three month's work produced only about twenty or thirty words to which everybody could agree. It was an example of vice-presidents, vice-chancellors and administrative staff sitting around arguing about commas and minor word changes. Exxon did exactly the same sort of high level study at the Executive level, and they only produced five hundred words in five years (about data on the international level)! It was a full time committee, too.

Getting dictionaries and getting definitions of words is a terrible problem, but essential. If we use the wrong word, we get the wrong answers, or no answer at all. Moreover, the use of an alias which appears correct may surprise the user. "How many persons in the database?" maybe should read "How many people in the database?": answer "0" instead of "5762"!

The use of directories tells where data is located, and some of its physical properties. Maybe you do not really want to know, and maybe you should not know, but the system must if you do not. For example, if you have a distributed system, and some of the data that you wanted is in the local data bank while some of it is in Los Angeles, you can expect a ten second delay when you ask the question. Knowing why might help; if yesterday you had been asking a question that seemed exactly the same but required only local data and received the answer immediately, you assume that now something is wrong with the system. Nothing went wrong with the system: you did not know that the data was located at a distance. Again, if you had known that these were high connection costs, you might decide you did not want to ask that question anyway! So, directories tell you where the data is located.

Now during the initial perception of need of a system, one must identify certain properties of the data. What data is needed? Does it exist? If not, go and collect it! If it actually exists, find out where it is and determine its characteristics. I call the mechanism for obtaining this information a catalog. But catalogs do not often exist; all too often, you cannot go anywhere to find the answers. But in some of the larger organizations, and EPA is typical of the sort I am discussing, you need to have catalogs of the availability of pollution data, and so on. In the Ministry of State and Urban Affairs in Canada (with which I was consulting a while back), they had exactly this sort of a problem. Often, they need to determine which library has what years of data on population in what region of Canada - and this is more useful, in fact, than having it sitting on a rotating disk close to the system.

So, the catalog components are items describing what and where: a uniform name, or standard name within the system, some standard or normality of definition which provides an exact meaning (to ensure that two people do not use the same word for different meanings), and any other name by which it is used (like synonyms, abbreviations, the names that are used in different programs). Next, we have qualifiers, like prefixing and suffixing (e.g., because the word date you only have to define day, month, year). However, week-end-date (i.e. with the prefix week-end-) should not be differently defined. So, all dates have the same properties, but the prefix or suffix is added to them. The other properties are the sources (places and availability) and the method of computation. There are many reasons why one must state methods of calculations. One of my favorites comes from Urban Affairs. "Quality of life" is something that has no standard meaning; therefore, we must know who calculated it and what sort of computation they made or the value is zero!

Now, the second type of element in a catalog is the logical characteristics; first recorded value sets, i.e., over what period of time, at what locations (e.g., depths in the ocean, if you're dealing with salinity). Next privacy requirements inherent with the data (if it has any), any structuring and associations between this data and other data, the units and dimensions of it (because we should not add meters to feet!), and any accuracy inherent in the way that the data was collected.

The physical characteristics involve types of storage media on which data exists (maybe just printed pages), formats (the grouping of elements and the needs for conversion) and the responsibility for collections and validation of the data. Again, remember that with data systems, the person who collects data is often different from the person who uses it. Thus we are forced to delegate responsibility and authority to the collection agent, and they must accept responsibility for providing high quality data (as needed). This is all too often not the case at present. Alternatively, the collector is interested in only a part of the data, but the other users are a long way away, so the collector tends to be sloppy in obtaining the extra data.

If we move over next to the second stage, which is the statement of requirements and design of the system, then we have identified that we need some sort of a system. We have probably also decided on an online system, and performed some thinking about organization for database, and have vested power in a mythical god (Superman or Wonder Woman) known as the DBA (Database Administration). In practice the DBA is not one person. In many organizations, it is a team, and often arranged as a hierarchical system. In this, the DBA function exists at many different levels within the organization, but is hierarchically controlled through the organization so that the lower level DBAs do not violate what the superior DBAs have stated as policy. The problem of implementing a good DBA function is Management (with a capital M!). Obviously, if you have policy at the top and you subvert it at a lower level, you have ruined the system.

What then does one do in order to be able to deal with these sorts of what I am now going to call Dictionary Systems? Well, dictionary systems generally have all of the attributes of a Catalog System but they must also have added attributes because now data is locally available and we practically need attributes like formats: the picture or physical representation of the data as it exists within the computing system. We also must know how data is produced for reports and is input into the system for new update: information that tells people how to utilize that system or how the system works with data as well as groupings by usage. If you know how data appears in reports, that represents an important need to group within the system, or it gives some concept of the way that data should be structured in order to store it in an effective or efficient fashion: Usage groupings, then to help things work well together, and the logical and storage structures, the way that data is actually stored. There are obviously different ways that people look on the data that they use. Some look on it hierarchically (in the same way that any sort of organization is structured), some look on it by relationships between various entities or objects, and these are then "networks", but all different methods are ways of modeling systems. The operational characteristics of data are attributes which again state who is responsible for it, who is allowed to do updating or is responsible for allocating; i.e., who can put in new inputs, and who is responsible for the quality control of the data. They also include the quality control aspects of the data that must be built into the system.

What I shall attempt in the next part is to suggest that if we state the quality of requirements in a dictionary, the reflection in the system should not be the responsibility of a programmer. In other words, it is important to take this problem away from the programmer and put it into the system; thus the system uses the dictionary concept to provide quality checking. Irrespective of whether the programmer writes code in a program, the system checks validity against the quality control statements as soon as that data is read in. Whether, in fact, that is done inside the dictionary or whether it is done inside the database management system, it should not be assumed that the programmers will remember to do the checking in every program.

As an aside: why are large scale programs never fully debugged? Our old friend Captain Grace Murray Hopper has a beautiful example of this. The

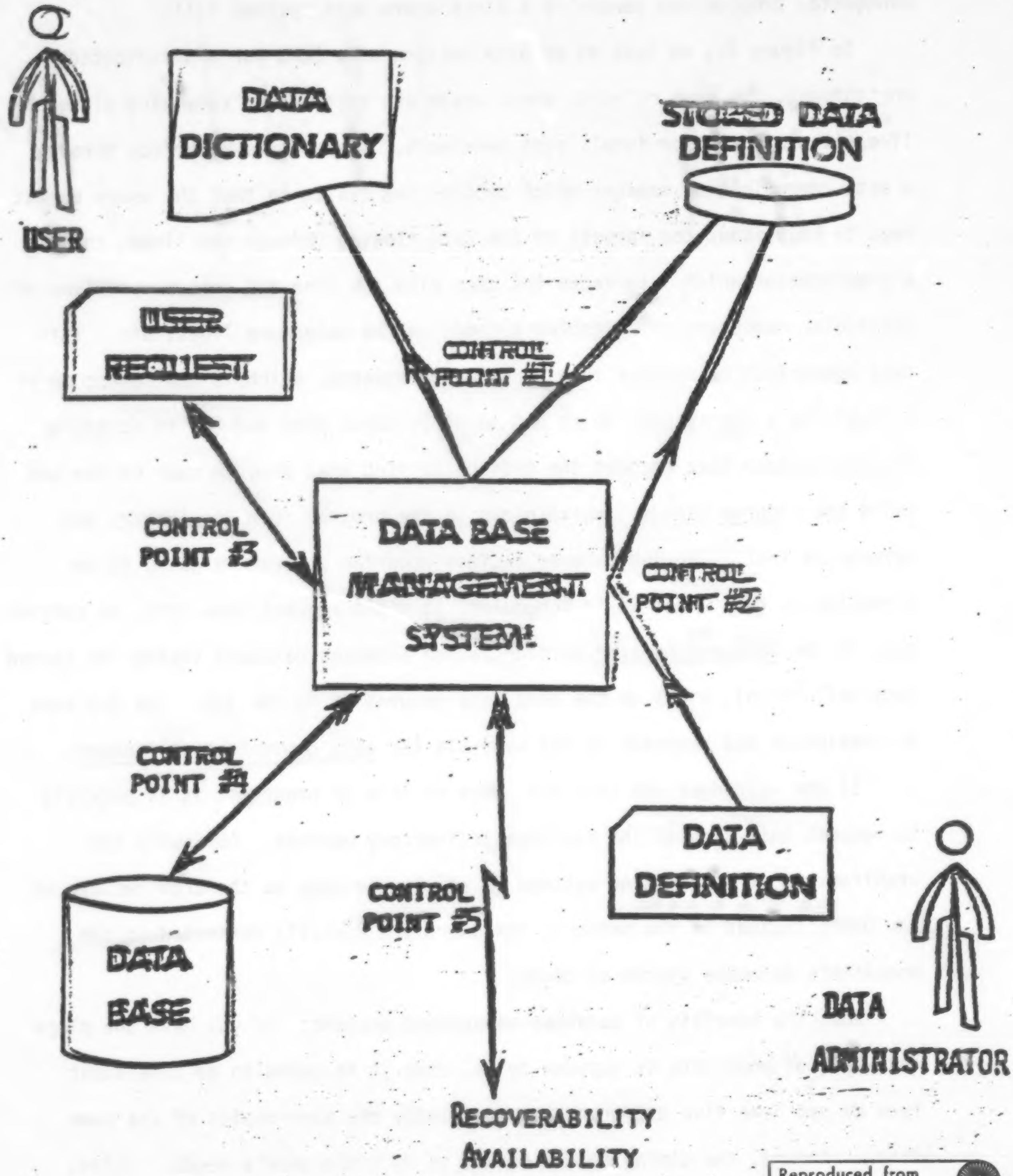
Univac operating system, EXEC 8, is large; if you started today to check out all of the various paths in this operating system, it would take you fourteen years (approximately) to finish the test. But the trouble is this, if you found a mistake (say seven years in), you fix it. What do you do? You cannot continue. You must start off again (because the fix may have introduced an error in something that you have already checked). So, you start again, and maybe in fifty or a hundred years get the operating system checked out properly: life is too short! It is better to have something that is not quite right than nothing!

Such problems also exist in database management systems because although they are not quite as big as operating systems, they are getting to be pretty big. Generally speaking, a new DBMS implementation costs something in the nature of ten million dollars on up.

Other aspects captured in a dictionary start with local responsibility, i.e., the privacy and security and authority aspects of it. Who assigns the rights for the privacy and security. This is not necessarily always the data administrator because that data administrator has to have fingers in every pie and presumably, therefore, has to again delegate responsibility. The integrity authority comes next - who says how often to dump or back up the file? This is extremely important, because we must not "lose" the file by some disaster - but how often must we copy it? Every night? If it is large and takes twenty hours, that will be expensive or impossible. Finally what is the need for keeping statistics on data usage? If we collect and store these statistics in the dictionary, the dictionary must run in conjunction with the DBMS (the system collects its own statistics and stores them over the dictionary). Then Utilities within the dictionary or within the DBMS can determine whether the system is working well or not.

Some other problems of the very large database environment include the assignment of value for data: is it collecting and storing? All too often, we implement systems that are not needed or that we cannot really afford. When we ask "What is the value of security?", we are faced with difficult answers. The bigger the systems get, the more expensive security becomes (security is the protection of privacy of the data). Integrity is also difficult to evaluate: it also becomes more and more expensive as we develop distributive systems. Quality: obviously the larger the systems, the more users, and the greater the probability of errors. Notice that security and integrity are front end loading on the costs of systems, and these aspects should be considered whenever dealing with large-scale systems. Figure 1. shows the principal control mechanisms of a DBMS. First, it controls the way the data administrator acts or it stops other people from not acting like a data administrator. The data administrator, for example, is allowed to make data definitions, which are the beginnings of the data dictionary. There are controls on who may provide updating, who is allowed to add, who is allowed to modify, then the whole of the stored data definition or the data dictionary. Next, the users can make requests of the system (either as clerks initiating transactions, which means that someone has written a program or as users providing query language statements by knowing enough about the language and the data that is stored). The system again has control because it can determine whether that user can make that sort of request, maybe even at that time. For example, I might very well allow some people to request the average salary, but I might not allow them to request the average salary of one person! i.e., we may allow a request for an average salary of at least five people.

There is one other important control point for database management systems: the fact that there is one place to tap for such important aspects of recoverability, availability and performance measurement.


USER
DATA DICTIONARY
STORED DATA DEFINITION
USER REQUEST
CONTROL POINT #1
CONTROL POINT #3
DATA BASE MANAGEMENT SYSTEM
CONTROL POINT #2
CONTROL POINT #4
DATA DEFINITION
CONTROL POINT #5
DATA BASE
DATA ADMINISTRATOR
RECOVERABILITY
AVAILABILITY
PERFORMANCE

Reproduced from best available copy.

FIGURE 1

All these control points are internal to the system; and, therefore, they are ones which are more difficult to subvert. Subversion of authority is a managerial problem and generally a place where most systems fail.

In Figure 2., we look at an architecture of a DBMS for a distributed environment. We have networks where users are sitting out somewhere along the line with teletypes or intelligent terminals. These users interface through a data communication manager which decodes the system so that the users do not need to know about the formats of the data flowing through the lines; this is a preprocessor which interfaces the user with the line and removes problems of particular computers or switching methods on the telephone lines, etc. This data communication manager then hands over requests, which is now looked on as a "job", to a job manager which has to worry about when and how to schedule. The job manager then decodes the message to find what program must be run and calls the program manager, which picks up the program from the library and returns control. The job manager decides when the program is going to be scheduled to run. When it is scheduled, it might request some data, so control goes to the database manager portion, which accesses database (using the stored data definition), picks up the data, and returns it to the job. The job runs to completion and responds to the user via the data communication manager.

If the user does not know the names of data or programs, it is possible to request this through the dictionary/directory manager. Obviously the architecture of distributed systems is really the same as the architecture of any DBMS; instead of the network, the users are directly connected to the monolithic database system of today.

Now, the benefits of database management systems: if you have one place where all of your data is suppose to be, then it is going to be consistent (you do not have five different but supposedly the same copies of the same data). Second, the timing can be turned to suit the user's needs. Third,

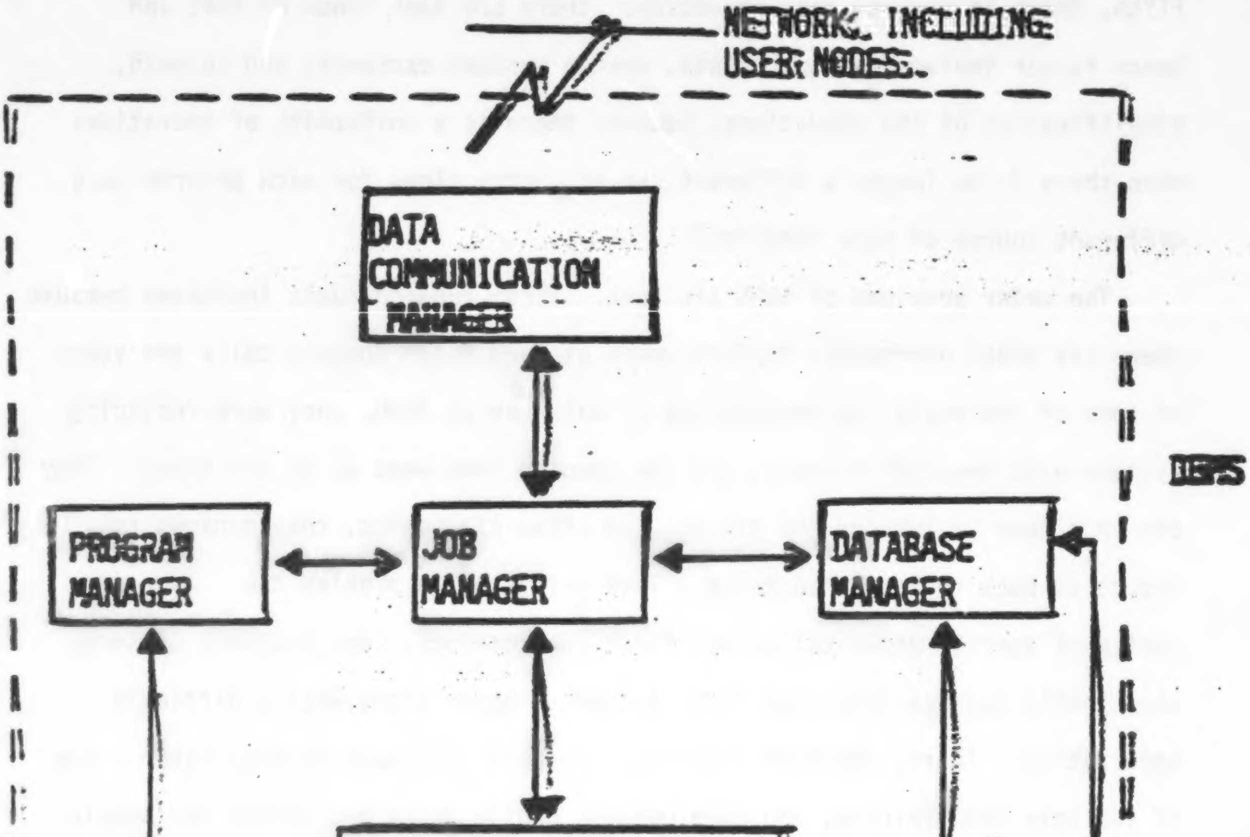

NETWORK, INCLUDING USER NODES.
DATA COMMUNICATION MANAGER
DBMS
PROGRAM MANAGER
JOB MANAGER
DATABASE MANAGER
DICTIONARY/DIRECTORY MANAGER
PROGRAM AND PROCEDURE
DATA DICTIONARY AND DIRECTORY
DATABASE
DATA DEFINITION

FIGURE 2

Reproduced from best available copy.

new functions can be provided quickly. Fourth, there is a simplified design with higher reliability with the higher level of interfaces and protocols. Fifth, there is program cost reduction: there are less lines of code and hence faster implementation. Sixth, easier systems exchange, and seventh, simplification of the operations, because there is a uniformity of operations when there is no longer a different set of instructions for each program (e.g. different source of tape handling).

The major problems of DBMS are next. First running costs increases because there are added overheads; Mayford Roark of Ford Motor Company tells the story of some of the early implementations of database at Ford; they were replacing systems with new DBMS Kernels, and the running time went up by ten times! They set up a team to improve the timing, and after six months, they managed to reduce it back to less than twice. This brings us to problem two: improper choice of storage organization and first applications. Bob Benjamin of Xerox says "build success into your first system" - never start with a difficult application! Third, improper training - I shall come back to that later - but if you have bad training, you have useless people doing bad things (or people not capable of doing their job properly). Fourth, lack of administrative objectives. Fifth, plan for conversion, the cost is not negligible. Converting a system to DBMS operation may be almost as expensive as starting over. Sixth, use a system design approach, but plan to implement piecemeal; the problems of DBMS implemented systems is not that they don't work, but that they don't work together. If you take several pieces of a jigsaw puzzle and try to put them together without the rest of the pieces, it is very likely that you won't be able to do it; this happens in systems that we design piecemeal. We need an integration plan with a proper design prior to starting. Seventh, look to cost distribution; it may be necessary to suggest a reallocation of costs so that people are not penalized for participating in an integrated DBMS environment

There are some questions which I think we have already unearthed. Is there any such thing as a perfect database management system? My answer is "no", because if there was, it would be tailorable to all persons needs: there is no such thing at the present time. Is there a difference between business and engineering in the area of database management? My answer is "no": engineering really does represent the application of a large volume of data in the same way that business does; nothing says that database management should not be used in engineering. There are certain programming language aspects that help engineers more than business people; e.g., we should not rewrite FORTRAN programs in COBOL and vice-versa. Thus, there might be interfacing differences. Standards - who cares? The answer is: you ought to. Validation security - are you willing to pay? The answer is that you should, because ultimately you are responsible for validity. If you have bad data, you will produce bad answers. If you have bad security, you will have problems in errors or data loss. Who cares about redundance? Well, somebody must be worried about it, or the system shuts down.

I have already discussed implementation productivity of DBMS. I just want now to discuss a tradeoff: between operation and implementation costs. It will cost more to operate a DBMS implemented system because of the overhead of the generalized system. However, it will probably cost you less to implement it, because the people will have less to code.

The second type of implementation productivity concerns the richness of database management; DBMS were written by people looking for a market; and therefore, DBMS may have functions that the users may not need but which cost space or time. However, DBMS implementers often provide easy add-ons that are compatible. Retrofit is expensive. It's always earth shattering; and therefore, it's often cheaper to work with DBMS.

The third type of productivity trade off occurs because of the need for education and a bigger staff. Somebody must maintain the DBMS and there is initial education of all staff.

During the design phases, the number of different concepts that are considered depends on the initiative of the team. Database management systems are loaded with "good ideas" and these deserve attention; by looking at the features, a better design may result.

As far as the operational phase is concerned, DBMS provide performance monitoring features which allow easier iteration towards a better system. They also provide good restart and recovery facilities. Again, by factoring these problems away from the program into the system, you pick up advantage on every program written.

Finally, in the maintenance areas, there is usually less maintenance because the interface is not so deep in the system. The DBMS is a higher level interface, and the DBMS vendor worries about operating system changes and the compatibility problems. Equally, in transferring from one machine to another, there are many software houses nowadays that are trying to provide uniform interfaces so that you can transfer (with them) to a new machine.

Database management happened: it happened in an amazingly short period of time. If we look at the history of data processing, we find that there really was no such thing as database management or even a concept of database management until somewhere into the very early sixties, though COBOL was a precursor to very many ideas that ultimately became database management ideas. In the early sixties, there was a lot of data system development in government areas, particularly in the intelligence agencies; the military, particularly the Army and Navy, in some of their developmental operations (again for logistic commands) were also early researchers in the field. These then provided the beginnings of database management; but the systems they were very inefficient: not cost effective, in any sense of the word. It took until the late sixties before the area started to be effective. 1967 was the time that CODASYL Systems Committee started to look into the area of database and tried to find what it was and where it was. By the early 1970's, in fact, it began to look as though DBMS were here to stay and that they were a good idea. People were beginning to experiment with database, and by 1975, we had really become DBMS "users".

Now, why? The answer was really that things really started to become cost effective, and they became cost effective because electronic equipment was dropping in price (radically!). About every seven years, we see a ten times hardware cost reduction. The cost of interfaces (input and output devices) were improving at about the same rate. Communicating systems were becoming viable. Time sharing "happened". In fact, DBMS were essentially, you might say, a molehill that suddenly started to expand due to the internal pressures of all of these advantages. The user aspirations, in fact, forced the system to "go", with the result that database technology was important; but it was oversold, as is every technology. It was oversold by salesmen; it was oversold by professors; it was oversold by everybody; e.g., I was tauting DBMS in 1970 as a good idea. Almost obviously, the consequences were: Systems were badly implemented, people did the wrong things with them, and the management of the concept was even worse, i.e., it was too much too soon.

Database technology does imply technical innovation, and everything that I shall say can apply to almost any technical change. You may have heard people say, "when we put in this piece of equipment into the factory, it changed the way we did everything, and everybody hated it. Everything that could go wrong did, and it took a long time before we managed to work it out". Technical innovation always causes or tends to cause problems; the bigger the innovation, the more problems. Reorganization is a good example of innovation (a managerial innovation), not always properly thought out, and often causing problems, worry, and contention. In general, under any change, the management must provide good support, or otherwise the normal operation becomes impossible. I have already talked about data administration. I pointed out that the Federal Government (at the topmost levels) are considering the need for Federal Administrators of data and, therefore, this suggests that there is a change which impacts all the way down to the lowest levels. There will, therefore, be a need for some restructuring, restructuring of the people, as well as restructuring of the way that they do things. Now, technical change, in fact, is affected by two factors. First, "What is the state of the enterprise at the time that the initiation of this innovation occurs?" and second, "How do the managers work?" Is the system in essence, autocratic? Or are there democratic means of expressing disagreement? Often, managers do not come into either category, but somewhere in between. The autocrats can generally push change, because they say it is going to happen, and they have "yes men" who will implement the rules. On the other hand, we have people who know how to "manage well" in the sense of being able to communicate. They can communicate the needs and because of communication, the needs occurs (all down the line). Everybody understands why and so, therefore, the implementation again goes well; but somewhere in between there are problems. Maybe the autocrat only manages to work for a certain period of time (before the revolution) - which introduces the next point: that the speed of introduction of the technology generally also effects the ease of the change on the corporate structure. Change can be evolutionary or revolutionary; Herb Schwartz talked about this idea yesterday. A revolutionary change causes all the turmoil associated with the revolution; so you cannot introduce a major technical innovation without total upper management support with middle management support as well as lower management support. Everybody who is affected must realize why; everybody has to realize how and everybody realize what!

I shall take two examples; the two examples show the introduction and problems of two different types of database management systems. There are both small and large database management systems. They were each developed to suit a certain set of needs, and your choice of system depends on which best suits your needs. At one end of the scale, there are the smaller systems and the more specialized systems. I shall now discuss two situations and the perfect way that each is supposed to happen. First we have the ideal single function case.

In such a situation, there is one principle function in the data system; e.g., a production environment which is not interested in further application areas such as payroll. We prespecify that we shall only use the DBMS for this operation, and this is our environment: we have totally circumscribed our job. But why is this helpful? It is helpful because we have no "cross-overs". You do not have one division or department who can come and update in our database; we "own" the data. It is our community of users who are being serviced by it. Thus we have localized committment; the people who are serviced and these that are servicing are localized. The system is not geographically distributed. It is not managerially distributed. In other words, there is no external community, and the users are essentially homogeneous. They are a small enough community that they tend to use the same terms. As an example, we do not need to define the word pollution because anybody using the database must know what it is. Therefore the need for a dictionary disappears (when you have a small enough community that talk the same language). They also tend to use words and apply procedures in the same way. They run the same sorts of program, or they ask the same sorts of questions. They do not have wide disparity between the way that they expect or anticipate using the system. The execution times are generally within a small range of one another. Now, this fact is not absolutely necessary; but it tends to imply homogenization.

The next important factor is that they all report to a common manager. Reporting to a common manager means that disputes are local: the two who disagree go to their manager and things are worked out quickly. Again, this reduces irritation, and there is an effective management system, with less rivalry and less need for resolution. Finally the programming is generally local - within one unit, and in some external department; they are communicating with programmers who are part of the family and know your sort of problems. Again, the need for the dictionary disappears because they know the terms that are being used within the database, or else there is a small dictionary of terms that they keep in their heads. Communication and naming conventions, generally, are simple. File management (e.g. the Mark IV system) generally are early DBMS; file management systems and the smaller database management systems have much in common, but database management systems have a few extras. Both use some sort of a stored data definition, which is the beginnings of a dictionary. Both generally provide some sort of systems support functions like restart-and-recovery, quality-of-performance measures, etc. They generally allow either programmer or query language interfaces, and they generally have aides within the system for the general creation and maintenance of the data. Now, typically, if you had such a system, you could talk about the implementation; and the implementation, generally only takes a short period of time. In such a case, you can bring the system in (with a little help from the vendor) and within a week or two, you have your first system working with a small group (of two or three people) who know how to use it and it is an instant success.

So, typically, I give this scenario. In 1974, someone in the data processing shop, or maybe someone who is a manager, has been to AMA workshop and comes back from it with the idea that "There is a good thing called database management and why don't we do something about it?" This manager talks at a general staff meeting, and one of the systems programmers says, "Gee, that sounds terrific", and is delegated the job of looking into the advantages and disadvantages of database management. Soon (in early 1975) the decision has been made to go to DBMS and one has been evaluated and selected. By the middle of 1975, (in less than six months) the whole system is working at the site (with vendor help). The group did not need a database administrator, because the function is taken over by that small community of programmers who are collectively the database administration. The assumption is that all the group are honest and professional: they need no policing. By late 1975, the whole system is working well with six application programs, and everybody is happy. There are many places where the situation is this good.

We now look at the ideal multifunction case. In the ideal multifunction case, we look at a bigger system. There is need for data to be shared across functional lines; e.g. there is not only production, but also the personnel data, and payroll too - all combined together. This allows operations in the engineering department to determine good production and scheduling. We now need data sharing, because different departments are involved. All these systems must be planned together, because otherwise there will still be three or four monolithic systems (which do not talk to one another), with no data sharing! Of course, the implementation can be staged - but within one plan. There must be top management commitment to this whole concept, generally to bring in data administration, set up a group as this function, and provide them with all the resources, manpower, and software devices that they will need.

In these circumstances, what do we need? We need a data dictionary, because now there are different environments; there is no longer a close community of users. We need to provide education for all the affected people and to have some sort of monitoring aids in order to be able to make sure that the DBMS is working properly. We must rid the comments of ownership and territorial claims. People have collected the data. They feel that it is theirs. They don't want to give it up; first, they might not wish to release information too soon (e.g., in making trial accounts); second they think that the people who are going to use it might update it in error. In other words, literally change the values so that there are errors caused by someone else, not in your control. Ownership concepts die hard!

Next, we have the concepts of privacy and quality or integrity. The database administrator must provide security and quality controls vested in the people who collect and develop the data. Finally there is the need for internal auditing. Internal auditing brings many advantages, not only from the financial standpoint, but also from a standpoint of quality because the internal auditors can and will determine why things went wrong. They will track back through the system to find when something got changed incorrectly; this immediately allows fixes to software (or people) if somebody made a mistake, or had no authority to change the data. There is a security violation. If the person had the authority but did it incorreclty, there is an educational violation (i.e. a need for explanation ...). If they keep doing it, fire them or give them more education. If the error occurred in a program, it can be further debugged.

In the ideal case then needs longer to implement. We start back in 1970 with a study team required to make decisions on use of a DBMS. The study team generally has four different sorts of people. There is generally a manager and three technologists, someone knowledgeable about database management and the problems of utilizing database management, and somebody who knows the actual systems that are running in the current operation: people who span the user and system environment, plus a manager to keep the whole thing fair. Note that if this group is successful, it becomes the nucleus of the data administration staff; they represent the types of technology needed for effective data administration. Within a couple of years, there were seminars available for management and the staff who are going to be affected by the DBMS, and a five-year plan had been set up. A dictionary has been started. Please note, the dictionary preceding the database management system. If you do not have a dictionary first, you have sown the seeds of trouble. By 1974, the dictionary has been implemented. The dictionary is made up from the inventory of applications, which may be very large. Some data dictionary system can load the data dictionary with the data definitions that occur in the original COBOL source program data definitions. Of course, this needs administrative resolution of conflicts, but this process allows for a "good start". By '74, a fairly large dictionary is, therefore, available and some resolutions had been made. By late '74, a decision had been made, and a database management system selected. In '75, the total system group understands the operation: it is "fully educated", a three or four month operation. In this six months period, the education process has produced programmers who know database management, but they have only written one toy program and still need to learn some internal problems: they are not yet really experienced. Most organizations find that it takes a year from the time that they initiate education on a large DBMS and the time that they have people being effective at writing programs for it. Now, commitment becomes the important issue. This is the commitment that we shall have all future systems implemented in a uniform way on this database system. Moreover, any major modification to old programs shall be converted to DBMS. By '76, the first operations were up and running. Others were in progress, and a switch over to online operation was complete. Going to an online operation also requires education, and knowledge by the programming and systems programming staffs. Now, because management was committed, they had provided the resources where needed and the database management staff had the tools that they needed: for example, in collecting monitoring and simulation data on the operation. They were also provided with design aids and report writers. They had initiated forms control (for updating the system) and the administration staff were well in charge. In '77 networking with the parent company was required. However, this produced little trauma because the various parts of the company (geographically dispursed) had been working together to ensure compatibility within the system. Finally, by mid -'77 everything is going well. Laison groups are working. Long-range plans have been accepted by upper management. Auditing is successfully implemented and, the system is accepted... but this effort took seven years!

Even in the places that show maximum success there is generally something like seven years lag. Richard M. Nolan discusses the four stages of implementation of systems. He talks about the implementation of large-scale computing systems within any enterprise and suggests that it takes seven years, which represents a "learning experience" within the enterprise. He also looked at the introduction of database and he found again that it introduced yet another "learning curve". Apparently, no matter what stage of learning, going to database management introduces another learning curve of five to seven year duration.

As I suggested earlier, this is a case of the introduction of innovation. Nolan had hoped to be able to accelerate the "learning time" by knowing the problems. But in an experiment, he found that this was impossible - apparently every company has to make its own mistakes!

Now we turn to the real facts of DBMS implementation; but it is often difficult to find the facts. They come from disgruntled employees who have left or else from friends that tell you, confidentially, the way it really happened. I have two friends who helped. One said, "Look - if you even write this up so that they guess that it is ours, then I may be fired". These messes are inflammatory! So, there's intimidation in talking about problems. I was talking to a man from Levi-Strauss who told me that he personally knows of twenty different one million dollar data processing fiascos in the last year.

I shall now talk about real word problems, and I shall try to give you illustrations of what went wrong, and why it went wrong. The mistakes are primarily managerial. These are technical problems, of course; but most of the technical problems only require work. Management problems require unsorting and incredible rework. What are the problems with real single functional systems? The first is just a pure "cussedness of life" problem: all too often, one finds that there is no such thing as a purely single function need - you are forced to cross lines. Somebody external to the section, on knowing that you have data, develops a need for it. The result is the need for retrofit: a slow and expensive proposition. The problems I can cite are true, but from many different places: e.g., you remember I said that the manager comes back from hearing a talk, has a staff meeting, and says to the people, "Looks like a good idea", and one of the systems programmers immediately says, "Hey, I'd like to do it". Now, you know that if that person is a first-rate systems programmer on your staff, you say, "No" (you cannot afford the loss of service). So, generally the job is given to the person who is not doing anything special (and who generally is not very effective). This one person that is the most unlikely to make a good job of it. In this case, she (it was a girl) probably would have made a mess of implementing a simple sequential processing system, and she produced a database disaster. It couldn't help being a disaster. The problem was one of commitment in getting a good staff.

The second error is to choose a bad application. If the first application takes too long or is not very interesting, there is no success criteria built into it. What is needed is a "flashy system" to show how good a DBMS can be. Preferably something which is online; something which shows a savings; something which is likely to produce satisfied users who will get up and say, "I couldn't have worked without it". Therefore, just choosing a good first implementation is important: it must serve as a selling tool.

Next: choose a good system design. If you take first or second generation systems and implement them on third or fourth generation hardware, you still have a first or second generation system; and all too often, that is exactly what is done. As an industry, we ignore redesign. The next problem is lack of communication. If a centralized database administrator concept is not implemented, there will be problems, even in small systems. I know of one example where they started without a data dictionary; but they found that they needed some method of recording names, and used a series of ordinary 2 x 4 cards sorted in a box on a desk of the program manager's secretary. It was better than not having any sort of data administration. (In essence, it was a very simple dictionary.) Most people who have started by saying "We don't need a data dictionary. We're too small." have lived to regret it.

The final point is the total instability of the functional systems. A friend on the staff of a large-scale manufacturer of chemicals told me that they had managerial decrees that they would be operation along lines only. The managers of the divisions decided, therefore, not to allow central database administration crossing functional lines. Strangely enough, about six months later, there was a company reorganization. Some divisions were joined, others divided. The result was a total change with need to reorganize all dictionaries - with more confusion and a decision to have central dictionaries after all. In other words, in this particular organization, they did go along functional lines; but the functional lines changed. As a result, the system was blown out of the water again. The problems, then, are that the controls are not properly enforced.

I know of one government agency who spent ten percent of their data processing budget on developing a system for five years. They have admitted at this point that the system doesn't work. It was badly planned. They brought in a top-level committee to look at it, decided that there was only one thing to do and that was to scrap the entire thing, change database management systems, bring in a new managerial staff and start over. Ten percent of their budget for five years is a fairly substantial investment. What they have done in that particular agency is to replace what were essentially worn-down technicians. They had become victims of the peter principle of promoting technicians to the point that they were no longer technicians and had become managers but they didn't know how to be managers. They are now replacing those technicians by engineers who know how to manage.

The second area is one of lower level management problems. Managers tend to optimize their position by expanding their power base or otherwise aggrandizing their position; this results in internal stresses in the management system; it means that people try to retain control; e.g. to retain the ownership of their data. Breaking down that barrier is extremely difficult. As an example, one chemical company decided not to implement an integrated IMS database system. They have five different functional operations; they do not have a centralized dictionary; and, in fact, they admit now that they are so badly confused that

Next, we see problems due to policy change in implementing DBMS. First, programmers resent their loss of responsibility and feel constrained by the effect of DBMS; they were previously essentially free to do any programming task in the way they like best, and, all of a sudden, the trap closes: they are no longer allowed to write data definitions and may lose the interesting parts of their programming.

Systems programmers equally resent the standards. They are losing the capability to be able to get into the system when they feel like it. In Equitable, they now enforce the rule that nobody changes that system without a write off at the vice-presidential level. There are many vice-presidents in insurance companies, but somebody in authority takes the responsibility for allowing change. The system programmer cannot shut down the system and "unravel a few pointers"; the vice-president authorizes the change and a member of the internal auditing staff checks the operation of the system programmer; then, all of the controls are reinstated.

The users resent loss of ownership and communication standards are broken. As an example, in a large manufacturer using IMS, a big simulation had been run to determine good parameters. They decided a five-K message region was necessary for communication based on response time considerations: i.e., for the communication region between the databases and the system. If they kept the message region at that size, they would get response in the five to ten second range. Along came a big new system that was being designed for the marketing department. It appeared to have no problems with region size; and, therefore, nobody worried about it in the programming staff until they were within about a week of putting the whole system together. Then they found that they had had poor control during design, and that the message region was going to have to be bigger than the standard. So, they went to the data administrator who said, "Too bad. You must correct it." So they went to the they have made a total reappraisal. They will keep going with the systems that they have already implemented, because they put a lot of money into them, but they now plan to implement a generalized dictionary. They will have to retrofit and they think that is it going to take them five years before they can recover from the mistakes that they have made over the first two years!

The third area is subversion of standards. If you do not keep your dictionaries up-to-date, if you do not make sure that people use your dictionaries, and if you allow people to write programs using names which are not in the dictionary, you have built-in failure because some program maintained will later be given the task of changing that program. The non standard names are not the common words of the dictionary; the person who wrote the program has now left, and the system is now no longer easily maintainable (if it ever was!); nobody is going to know really what it is all about; so, therefore, it is going to cost everyone in retrofitting to the standards.

As I have already said, pricing structures generally have to be changed in order not to penalize people who have become accustomed to sequential processing costs. The average user needs to have price stability.

The mistakes and problems are more managerial than technical - personal communication problems, and lack of understanding of the effects of a simple subversion. Management can often make exceptions, but they have to realize the effect of a small subversion; which may "cast a long shadow". Mistakes, lack of commitment, and policy changes when management changes build in problems. Until managers understand the problems that they can generate by changing policy on data, they are dangerous. We need commitment at the top level. A DP management must tell their top manager, "You must realize what you do by making or breaking our rules". Maybe consultants can help! They can tell top managers the facts, and they may believe it.

vice-president for marketing, and said: "Sorry! Because of the data administration, we shall be late." The vice-president of marketing said "I'll fix that." He went to see the president. The president said: "Who does this data administrator think he is?" Data administration was forced to back down, and on putting the new system in, the whole system went "flat on its back", with fifteen to twenty second response time. Everybody hated the system. The good part of this story is that the major user of the system was the marketing vice-president's office and they felt the impact of their own stupidity - until they returned to the original standard.

In implementing a distributive system, problems often arise. I have two stories, in fact. One is a military system. The logistics group have a perfectly good database management operating: the "brass" have a small facility and use index sequential access methods. The "solution" was to build an index sequential system on top of the database management system in the logistics groups. Thus, everytime the logistics group perform an update, they have to update their database (in the database management system) and the top brass data (in the index sequential system). A good example of poor planning and "the tail wagging the dog": they had to cobble together a system because they had never planned it right. The second illustration involves field offices. The Department of Agriculture field offices had been providing formatted data to a permanent reporting system. Recently, they have discovered in headquarters that they need field office data which is not regularly reported. They need to put up a dictionary, but the result is confusion in the field offices: almost everyone had a different system. It will take a substantial retrofit to be able to get that extra data. Now, they have realized that the data dictionary should be properly hierarchical instead of just central.

I have already talked about problems of interfacing and of responsibility for auditing. Criminals usually help to "solve" this problem - by proving that it hurts in the pocketbook, and gradually people begin to worry. One man was hired by a friend of mine to be a Data Administrator (on the West coast). My friend was later talking with another friend from one of the (large) local banks; the banker said: "I hear that you have hired Charlie so-and-so to be your Data Administrator". "Did you know about us getting rid of him?" "Well, it turned out that this guy ripped us off a quarter of a million dollars; but, being a bank, we were scared" (to let their depositors know about it). "We told him we wouldn't prosecute provided he left." Now my friend is in insurance and found that his trusted data administrator was a crook! He could not fire the DBA because he received the information confidentially; he could never get anyone to be a witness in court on a wrongful dismissal case.

Finally, in summary, what does history do for us? History provides us with experience. The trouble is that old experience may be wrong experience when you switch into database: it may be necessary to unlearn in order to move to database. Next: democracy is a great idea, but not in a data processing environment; there has been a tendency for people to have rights to data, and to constrain or restrict the operations upon their data; we advocate database administration, involving integration. Again, this causes problems whether we like it or not. Another is a fact of life. Data processing has always been optimistic, saying, "we can do it"; and users have been burned. We need to rebuild confidence. Finally, how do we implement the data dictionary operations and naming problems in a large organization? The answer is "with extreme difficulty, if ever". The bigger the organization, the more the problems.

Conclusions are: we need database administration at high levels, probably hierarchically structured within the system; we need auditors within the database administration staff; we need managers (not out-of-date technicians) looking after the process; integration has to be planned from the very beginning; politics and poor management are the worst problems; a data dictionary is essential, if you do not start off with it, you will become more and more confused; the database administration group needs resources: personal and tools; the DBA must be at a high enough level to command respect; education is obviously of paramount importance; the first implementation ought to look good, so start with success. I have tried to sum up some of the problems that face you as managers in the implementation of what is a promulgating problem. Systems that we are developing insidiously move into the whole of any organization; they need heavy managerial control, authority put where it is necessary or we may as well give up before we start.

Question: It seems to me that the pictures that have been painted in the last couple of days is that your staff is going to grow under DBMS. It's going to take seven years to do it through which time hardware and software packages which you are trying to implement will become obsolete. You've just got a high risk atmosphere. It sounds like it's going to cost more. I'm questioning why anybody would want to do it.

Sibley: First of all, it will not take seven years to get started. The seven year time-frame supposes that the first two years are evaluation (before you get committed); during the other five years, you are actually running after about a year. So, you see the beginnings of payoff from a year on, though full payoff may take five years. Now, this is not really a bad risk situation from the standpoint of payback. Second, the question about the size of staff. Sure, you have staff increases but is it a matter here of whose risk are you taking? Is it your risk? Is it a risk to the organization? Who gets most out of what? Now, unfortunately, as I suggested, organizations tend to suboptimize. Managers tend to suboptimize their share of the organization. There is probably a saving in some other area because of your system, the budget goes up in your area, but it usually doesn't go down in the area where the saving occurred. As an example, Chevrolet Engineering put all their design data into one set of files. There were two advantages. First, all the data used to be scattered around the company because they had different engineers in different places, generating mountains of paperwork which was distributed to many locations. Because local agencies had up-to-date data, the production start-up was easier, saving thousands of dollars. But second, because the old manual system had taken weeks to process changes, the procurement cycle was longer and there were costly cancellations. The reduction in cancellations paid for the system! But if you save a department several hundred thousand, that doesn't mean that they let four or five people go. They are normally put onto doing something else. So, now we have a philosophy problem. What are you doing? Are you making a particular system better? In some sense, we're making the whole system work better. We're presumably providing a better service; and therefore, presumably we are now making for total system which is better for the total environment.

Question: You don't know of our situation, but I think I can comment on a couple of things that might prove fruitful. One is that in comparing the risks of going from index to database management with that of going from nothing to index sequential, the risks are not all that much greater. So assuming you had an operational system, it would be a much greater risk; but in your case, I'm assuming that you are going to have to pursue aggressively and in the next sequential system, get what you want. Point two is there is a good probability that what you are going to need to do cannot be done by index sequential and only database management will suffice for information needs. I offer these two points for discussion.

Sibley: If you have got an operational environment which includes System 2000, you don't have to worry about some of your staff. They are being provided by the main office. Then, the incremential costs to a user community is not high. You already have a group of people who understand the user requirement and are able to produce some user programs. You are piggybacking on the things that other people are providing.

Question: I can see the application of this to what you might call management data; but I'm concerned with setting up some sort of data management system. I'll be very careful not to call it a database management system effort for a lot of scientific data and most of the work will be done by people who are scientists or engineers and who either can do FORTRAN themselves or have people readily available who can do COBOL. It's a small number of people in the database who we would hope to have exhausted within two or three years. I guess I came hoping to get information that would help me to decide something about what kind of data system to go to. I guess I've concluded that database is probably not appropriate.

Sibley: Sight unseen, I can't answer your question. If you brought in a consultant, you could get a better answer. It takes time to find out what is the problem. Fairly obviously, if you have people who can do FORTRAN and you have a well-structured file system, and you've got people in place, and you've a file of data, then maybe you're working as efficiently as you should (for the moment). If you find that you have got a lot of ad hoc requests and your FORTRAN programmers start to get loaded down, and you either have to increase your FORTRAN programmers or your users are being upset (or whatever), or you find that what is happening is that you are starting to get more and more requests which require just a small amount of data out of a very big file and you've been doing sequential run through, then it's going to start costing you money doing sequential run throughs to pick up two or three bits of information. Then you see the need for something to do things better. Now, I don't like to think of System 2000 as a "glorious access method"; but it has an access method built in. Moreover, it has its user language which allows ad hoc questions to be posed and often the users can write these high level requests (with the caveats that they could ask the wrong question). This does represent a reduction in time (for the engineer to write the query); certainly it is cheaper than writing a FORTRAN program, and therefore, it's a question of whether you are making them more cost effective. You can only find this out by examining a broad spectrum of what you do and saying how much is it costing.

Question: It's my feeling that if I were to do that and take the cost of doing that and compared any savings by going to the better method, it would cost me ten times what I had saved to make the savings.

Sibley: Well, in that case, it's not worth going to database! One time we did a time-in-motion study of how nurses work and we found out by rearranging a fairly large bed ward, that we could save nurses time. (This was a PhD thesis, the typical sort of stuff that you do around a university). The research student was terribly happy that he could save nurses time until I said, "Look, how much time is it?" It averaged out to be about ten minutes a day. I said, "What are they going to do with the time? Have a smoke? Does it really improve patient care?" I think yours is the same question. You must really identify your savings and if you're not saving anything, don't do it; however, if you have work to do then your engineers have time to perform, and you know you're not going to get any lines, then you need to be more effective with the people you've got.

Question: I can find lots of FORTRAN programmers but I can't put more than twenty-four hours into one day. It's going to cost me two-man weeks to decide which is the best system and if it saves five-man years, I still can't afford it.

Sibley: In that case, you bring in a consultant. I'm not trying to sell anyone. Seriously, if you think that there is an advantage to it, don't neglect it.

Mr. Willis Greenstreet
Management Information & Data Systems Division
Environmental Protection Agency
Washington, D. C. 20460

As Ed was talking, I was reminded of the old Pogo story. We are all managers in this room and we have found out that the biggest reason for failures of systems are managerial failures. I was reminded that Pogo said once: "I have met the enemy and it is us!"

We've heard, I think, an excellent seminar in the last day and one-half, frankly exceeding my expectations. I think we have given a lot of what we advertised we would give you. That is, the information has been presented for managers who are not data processing professionals but who have to approve and use systems. I think you have received some damn good information on how to procede from this point. We've been somewhere between the toe and the ankle in getting our feet in the water with database management, even before I came into this job.

Somewhere in early 1974, the agency selected System 2000 as a basic database management system for both computer centers. As Don mentioned briefly yesterday, we have a variety of file management systems starting with Mark IV and moving up the scale through Easytrieve and so forth. Now, we are experimenting with DMS 1100. There are some pushes for us to go further perhaps; but I'd like to just take a moment to discuss them.

As you know we have two computer centers. One is supported by an IBM 370/168 operated by COMNET, called the Washington Computer Center (WCC). The one here is an UNIVAC 1110 which is of a comparable size operated by SDC/ISI called the National Computer Center (NCC). We have in those two centers those packages I mentioned. Also, we are experimenting with data management package called INFORM 4 on the DEC PDP 11/70 in three regional offices and three laboratories. The basic package is being tested now. It seems to have some capabilities that are useful on a small computer.

I was also very interested in hearing Dick Nolan mentioned. As far as I'm concerned, Dick's philosophy is guiding data processing in EPA. We're a firm believer in the four stages of growth. Naturally, we identify ourselves as being stage 3. Whether we're there or not, who knows? I also have those reprints available. I buy them by the hundreds and distribute them to people who are interested in database management or the four stages of growth. If you contact me, I'd be delighted to send them to those few who haven't received them already.

The last thing that I want to mention is the fact that those who want system consulting of one type of another, whether it is a question of what to do for a small laboratory or even a question of what the merits are of a large-scale database management system such as IMS, I would suggest that you start with a phone call to either Sam Brown or Neal Haley; we'll try to provide a contractor resource for you to begin work. I know that Sam through his contractor ISI is acquiring some System 2000 personnel on a direct hire basis. We're trying to get that available in the Washington Computer Center too. At this point, UNIVAC is supplying direct support on DMS 1100. I assume we can use this support for a while until business gets too good (or not good enough) since they provide the services on a bundled basis; that is, everything comes in a basic price and they don't charge extra for freebies.

I feel like that we are in an evolutionary process with database management. That's basically how I try to approach anything, particularly data processing - very slowly and very gingerly because I have had some things blow up in my face. I still have some scars from other incidents.

I recall very vividly a staff meeting I attended. I believe Ed Rhodes was at this same staff meeting with Howard Messner back in September of 1971, which was the first one that I ever attended. For those of you who did not know Howard Messner, he was about the size of Neal Haley but he talks a lot louder. The main topic of discussion that day was the blue book. I don't know if you people in the field have heard of the blue book but that rascal has caused us more grief than anything else in this Agency. It's a summary of personnel statistics and it's never been right; it's still not right today. I'm sure Ed has had some fainting spells over it. At any rate, we went through a feasibility study. General System design then went through a detailed system design which is basically what you saw today as a pilot and I feel sure that we are going to go on to a database management system. We haven't sat down with the personnel people and talked about which system but I suspect that they are going to look for a recommendation from my division.

I think that there is another major system in the operational area that will be implemented using a DBMS. Again, I don't know which one it will be; but the water supply system is headed in that direction. In this system the violations and variance and exemptions for each state's water supply (could be thousands) are reported and massaged. Hence, there will be a lot of regional headquarter use of that system.

We're also involved in a so called 'systems audit' with the Office of Pesticides. It appears that they are going to be very interested in a database management system for their many product and registration files. It is just in the general feasibility stages at this point.

We have several smaller ongoing systems. As far as I know, these are all on System 2000. At this point, the Energy Data System at OAQPS, which was discussed briefly yesterday, is probably the largest system. There are a couple of management systems in the industrial lab here. Bio-storet, in its feasibility stage, was done on System 2000 as an R&D project. Whether that will continue as System 2000 or be converted to a Storet format is yet to be decided because a decision has not been made as to whether Bio-storet is going to be an operational system or not. There is a lot of political pressure in both directions on that.

The state of Wisconsin put up something called WINSEP which was an environment management system similar to the ill-fated GPSF. Unfortunately, for the state of Wisconsin, this one worked. It was on a smaller scale and was a classic story of the first implementation. It was very inefficient. MRI, who is the owner of System 2000, came in with some of their hotshots, and with minor alterations converted it to a very efficient system. Now, we are trying to find out exactly what happened for our own benefit; but it worked well. It's exceeded my expectations. I thought it was going to be a bomb. The reason we were involved in it was that they wanted to put it up on EPA's computer and use System 2000 so that they wouldn't have to buy the package.

Finally, the New York region brought up a data system for their own region on System 2000. It's a little inefficient but I think it fits the general description which has been given for the last day and one-half. It was operational very quickly but it would need to be refined and tuned if it was going to go into full production.

That's a review of what is going on in database management, particularly in the S2K area, and you saw the demonstration on DMS 1100 yesterday. I'm sure, seeing the faces around the room, that no one is going to be leaving here with a passion for DBMS. That was not our mission. Quite often, in data processing things are oversold. I think we have tried to caution you as much as possible. I'm really pleased with the emphasis of the speakers on the planning and managerial control and constraints because that's exactly what determines the success or failure of a system. In particular, it seemed that it has always been the case at EPA. Whether it be database management, index sequential or anything else, I think that the successful systems had management involvement from the beginning. And I think the lessons are only reinforced in using database management.

That's all I have to say today and I want to thank you all for coming. I think most people have already agreed it was worth the time. Thank you.